T. MUREȘAN
C. STRUGARU
R. STOINESCU
E. PETRIU

MICROPROCESORUL 8080 ÎN APLICAȚII

EDITURA FACLA

T. MUREŞAN, C. STRUGARU, R. STOINESCU, E. PETRIU

MICROPROCESORUL 8080 ÎN APLICAŢII

Coperta : NICOLAE SÎRBU

TIBERIU MUREȘAN
RADU STOINESCU
CRIȘAN STRUGARU
EMIL PETRIU

MICROPROCESORUL 8080 ÎN APLICAȚII

EDITURA FACLA
TIMIȘOARA, 1981

PREFAȚĂ

Industria din țara noastră produce în prezent echipamente bazate pe microprocesorul INTEL *8080. Dintre acestea se remarcă calculatorul FELIX -M18, echipamentul de automatizare ECAROM 800, automatul programabil AP 117. Ele reprezintă un pas important în direcția elaborării de noi tipuri de microcalculatoare, de echipamente destinate automatizării complexe, inclusiv prin introducerea roboților industriali, conform sarcinilor stabilite de Congresul al XII-lea al P.C.R. In acest scop baza de componente se va dezvolta prin realizarea în țară a circuitelor integrate pe scară largă între care microprocesoare și memorii semiconductoare.*

Gama de aplicații a microprocesoarelor este foarte largă. Este firesc ca industria să pună pe primul plan producerea de echipamente care să corespundă cerințelor impuse de aplicațiile complexe. Numeroase și de asemenea importante sînt și aplicațiile mai simple. În aceste cazuri adesea se dovedește mai economică și mai fiabilă soluția de a proiecta și realiza un echipament specializat, față de soluția folosirii unui echipament cu destinație mai generală, pe care îl produce deja industria, chiar dacă acesta este astfel conceput încît ar permite o anumită adaptare la cerințele aplicației respective.

Folosirea microprocesorului conduce la simplificări importante în proiectarea unui echipament. În același timp însă proiectantul este pus în situația de a face față unor noi aspecte. El trebuie să decidă, de exemplu, dacă o anumită operație sau comandă este mai bine să fie realizată printr-o schemă electronică (prin „logică cablată") sau prin intermediul unor programe (prin „logică programată"). Această legătură foarte strînsă dintre partea de circuite și partea de programe pune probleme noi pentru proiectantul cunoscător numai a metodelor tradiționale de analiză și sinteză.

Avînd în vedere cele de mai sus, o condiție a progresului rapid privind extinderea ariei de aplicații a microprocesoarelor este și existența unei literaturi de specialitate pentru pregătirea corespunzătoare a unui cerc cît mai larg de tehnicieni. Lucrarea de față vine să completeze lista materialelor publicate pînă acum în limba română cu referiri la microprocesoare.

Obiectivul principal al lucrării este să prezinte, într-o formă cît mai completă, un număr de aplicații ale microprocesorului INTEL 8080, aplicații relativ simple dar care pot constitui un punct de plecare în abordarea altora mai complexe. Prin modul de prezentare lucrarea nu constituie un material de inițiere. Pentru a

putea înțelege problemele expuse cititorul trebuie să stăpînească bine modul de organizare și funcționare a microprocesorului INTEL 8080, a celorlalte circuite LSI din această familie și să posede un minim de cunoștințe privind programarea în limbaj mașină. Pentru toate acestea se recomandă ca material de bază în limba română lucrarea prof. univ. dr. ing. IOAN DANCEA Microprocesoare. Arhitectură internă, programare, aplicații, *Editura Dacia, 1979.*

După prezentarea pe scurt a unor probleme de ordin general în primul capitol, în următoarele două capitole se descriu două variante de organizare a unui microcalculator bazat pe microprocesorul 8080. În ambele cazuri microcalculatorul s-a considerat că face parte dintr-un sistem, înțelegînd prin aceasta că este în legătură cu un număr de dispozitive de tipul intrare-ieșire. Interfețele cu aceste dispozitive sînt prezentate în detaliu și în strînsă legătură cu microcalculatorul la care sînt atașate. Există posibilitatea ca numărul acestor dispozitive să fie extins păstrînd aceeași organizare pentru microcalculator. Din acest motiv astfel de sisteme sînt cunoscute și sub numele de sisteme de dezvoltare. În capitolul patru sînt descrise circuitele de interfață cu alte tipuri de periferice numerice iar în capitolul cinci circuitele de interfață cu unele periferice de tip analogic. În unele cazuri descrierea dată nu se referă la unul din cele două microcalculatoare analizate în capitolele doi și trei, dar cititorul va deduce ușor cum poate fi făcută această legătură. Avînd în vedere că microprocesorul 8080 manipulează datele pe lungimea de opt biți, s-a considerat util ca în capitolul șase să se arate modul de efectuare a operațiilor aritmetice fundamentale pentru operanzi cu lungimea de 16 biți în reprezentarea complement față de 2. În capitolul șapte sînt discutate cîteva elemente legate de comanda unui robot de laborator.

Aplicațiile descrise în această carte au fost realizate și verificate experimental. La elaborarea lor s-a avut în vedere, acolo unde a fost posibil, folosirea componentelor și echipamentelor produse în țară.

AUTORII

CAPITOLUL I

ASPECTE DE INTERES GENERAL ASUPRA OPERAȚIILOR DE INTRARE–IEȘIRE

La calculatoarele electronice dar și la echipamentele complexe de automatizare, problema modului în care se pot realiza operațiile de intrare/ieșire prezintă o importanță deosebită. Prin operații de intrare/ieșire se înțeleg acele operații prin care calculatorul „comunică" cu exteriorul. De exemplu, calculatorul va realiza operații de intrare atunci cînd va prelua un program perforat pe cartele și va realiza operații de ieșire atunci cînd va tipări rezultatele pe hîrtie la o imprimantă. Un alt exemplu. Un calculator conectat la un proces va realiza operații de intrare atunci cînd va prelua valoarea temperaturii din cîteva puncte diferite ale unui cazan și va realiza operații de ieșire atunci cînd va comanda închiderea sau deschiderea unei anumite vane. În primul exemplu legătura cu exteriorul s-a făcut prin intermediul lectorului de cartele și a imprimantei, care joacă rolul de echipament periferic. Viteza de lucru a acestor echipamente periferice, pentru că conține piese mecanice în mișcare, este foarte mică comparativ cu viteza de lucru a calculatorului. În al doilea exemplu legătura calculatorului cu exteriorul este mai puțin evidentă. Valoarea temperaturii din cele cîteva puncte, probabil că va fi preluată direct de la niște convertoare analog/numerice sau de la un sistem de achiziție de date. Viteza de lucru a unor sisteme de achiziție de date poate fi comparabilă cu viteza de lucru a calculatorului. În ceea ce privește închiderea sau deschiderea vanei, nu calculatorul va efectua nemijlocit această operație. El va indica, unui element intermediar, de exemplu unui servomotor, sensul de rotire al vanei și momentul cînd această operație trebuie să înceteze. Este necesar ca într-un anumit fel calculatorul să urmărească tot timpul poziția vanei pentru a deduce cînd aceasta a ajuns în poziția dorită. Din cele de mai sus rezultă și diversitatea foarte largă a situațiilor concrete în care se pune problema transferului de date între calculator și echipa-

mentele exterioare acestuia. Din acest motiv se consideră utilă o analiză pe scurt, la modul general, a acestei probleme.

Operațiile de intrare/ieșire se pot realiza prin unul din următoarele trei moduri [1, 2, 3, 4]:

— transferul datelor prin program;

— transferul datelor prin intermediul întreruperilor;

— transferul datelor prin acces direct la memorie (Direct Memory Acces, DMA).

1.1. TRANSFERUL DATELOR PRIN PROGRAM

În cazul acestei metode operația de intrare/ieșire este comandată și precizată în timp de către un program de genul celui de mai jos:

X TEST DP1 Verifică dacă dispozitivul periferic *DP1* este pregătit pentru a transfera date. Dacă da, omite instrucția următoare;

SALT X DP1 fiind ocupat salt la adresa *X*;

TRD DP1 Trimite un octet la perifericul *DP1*.

Unei mașini de scris, cu viteza de tipărire de 10 caractere pe secundă, îi sînt necesare 100 000 de microsecunde pentru tipărirea unui caracter. Dacă calculatorul s-ar ocupa exclusiv de comanda mașinii de scris, atunci în cel mult 50—100 de microsecunde ar pregăti următorul caracter ce trebuie tipărit și restul timpului ar rămîne inactiv în bucla de așteptare. Dacă se consideră un echipament periferic rapid, de exemplu discul magnetic, atunci situația se inversează. Un disc magnetic poate transfera date cu o rată de transfer de 100 000 octeți pe secundă, adică un octet la 10 microsecunde. În acest caz rezultă că calculatorul este prea lent pentru că nu poate prelua datele în ritmul în care acestea vin.

Atunci cînd sînt mai multe periferice lente calculatorul realizează operații de intrare/ieșire cu fiecare dintre ele, folosind un program conceput anume pentru acest scop. Pe baza unei liste a tuturor perifericelor, calculatorul se oprește asupra primului din listă și-l interoghează dacă are nevoie să desfășoare operații de intrare/ieșire. După ce termină cu primul trece și interoghează pe al doilea periferic din listă ș.a.m.d. Calculatorul deservește perifericele secvențial și după același interval de timp. Este cazul în care perifericele apar ca avînd aceiași prioritate. Într-o altă variantă se acordă unor periferice o atenție mai mare în raport cu altele, în sensul că sînt deservite mai frecvent și un timp mai îndelungat. Acest lucru se poate realiza, de exemplu, prin plasarea aceluiași periferic în listă de mai multe ori.

Procedeul de transfer a datelor prin program este cel mai flexibil dar și cel mai lent și nu este recomandabil pentru perifericele rapide în regim sincron. Calculatorul pierde mult timp cu interogarea circuitelor de interfață cu perifericele. Prezintă însă avantajul că operația de intrare/ieșire se face la inițiativa calculatorului și este sincronizată cu programul.

Pentru a simplifica circuitele de interfață, atunci cînd proiectantul decide a realiza operațiile de intrare/ieșire prin procedeul de transfer programat, există circuite specializate în familia circuitelor anexă la microprocesorul 8080. Astfel, circuitul USART 8251 este destinat pentru transferul datelor sub formă serie în regim sincron sau asincron. Interogarea lui 8251 de către calculator se traduce prin preluarea de la nivelul circuitului 8251 așa numitei informații de stare. Prin cercetarea acesteia calculatorul va afla în ce situație se găsea în momentul respectiv circuitul 8251. De exemplu, calculatorul ar putea afla că precedentul caracter a fost transmis pe linie și că 8251 așteaptă de la calculator un nou caracter pentru a fi transmis. Circuitul PIO 8255 este destinat interfețelor unde se impune ca transferul datelor să se facă în paralel. Ambele circuite au mai multe regimuri de funcționare și alegerea unui anumit regim se face printr-o așa numită operație de programare a acestor circuite [4].

1.2. TRANSFERUL DATELOR PRIN INTERMEDIUL ÎNTRERUPERILOR

Procedeul anterior este nesatisfăcător pentru cazurile în care perifericul impune ca operațiile de intrare/ieșire să se realizeze neapărat în anumite momente, aceste cerințe apărînd însă la intervale de timp întîmplătoare. Pentru a satisface această restricție, prin procedeul anterior, calculatorul trebuie să rămînă tot timpul într-o buclă de program în așteptare pentru a sesiza cererea de intrare/ieșire emisă de periferic. În astfel de cazuri, cînd perifericul dictează momentul în care să aibă loc operația de intrare/ieșire, se face apel la posibilitățile de întrerupere ale programelor în curs de execuție. Perifericul va lansa o cerere de întrerupere, numită întrerupere de tipul intrare/ieșire, în urma căreia calculatorul va suspenda momentan programul în curs de execuție și va trece la o subrutină prin intermediul căreia va realiza transferul de date cu perifericul în cauză. La terminarea transferului calculatorul va relua execuția programului din punctul în care a fost întreruptă execuția lui.

La necesitatea acestui procedeu s-a ajuns plecînd de la ideea că dacă operația de intrare/ieșire nu s-ar realiza în momentul cerut atunci s-ar pierde datele. Metoda poate fi însă folosită și în cazul echipementelor care nu fac parte din această categorie. Reluînd exemplul cu mașina de scris, tipărirea prin pro-

cedeul întreruperilor ar decurge în felul următor. Calculatorul execută o subrutină anume prin care un caracter se trimite pentru a fi tipărit, după care își reia execuția programului principal. Cînd mașina de scris devine disponibilă pentru a tipări un nou caracter, lansează o cerere de întrerupere către calculator. Acesta va achita cererea de întrerupere către calculator. Acesta va achita cererea de întrerupere, în sensul că va suspenda momentan execuția programului principal și va trece la subrutina prin care trimite un caracter spre mașina de scris. După ce a trimis caracterul calculatorul își reia execuția programului principal. Pentru caracterul următor procesul se repetă. Procedînd astfel calculatorul este folosit mult mai eficient, mai ales în cazul perifericelor lente.

Circuitele USART 8251 și PIO 8255, amintite mai înainte, pot fi utilizate și în situația discutată întrucît sînt prevăzute și cu posibilitatea de a lansa cereri de întrerupere spre microprocesorul 8080. Acesta dispune însă de un singur semnal de intrare (INT) prin care se poate indica apariția unei cereri de întrerupere. Pentru a facilita recunoașterea perifericului, ce a lansat o cerere de întrerupere, în cazurile în care mai multe periferice pot face acest lucru, precum și pentru rezolvarea mai comodă a problemei priorităților, sînt destinate circuitele PIC 8259 și 8214 [1 ; 4]. Se face mențiunea că programatorul înainte de a valida influența semnalelor de înrerupere asupra microprocesorului 8080, prin instrucția EI, el este obligat să prevadă în program o inițializare a indicatorului de stivă SP. În caz contrar conținutul lui SP rămîne necunoscut pentru programator. La apariția unei cereri de întrerupere conținutul numărătorului de program (*PC*) se memorează la o adresă indicată de *SP*. Or dacă conținutul lui *SP* este necontrolat de către programator, memorarea ar putea să se facă într-o zonă din memoria *RAM* unde informația existentă acolo nu este permis să fie modificată. Dacă operația de memorare ar urma să se facă într-o zonă ocupată de memoriile ROM, evident că acest lucru nu se poate face și în consecință revenirea corectă în programul principal este compromisă.

1.3. TRANSFERUL DATELOR PRIN ACCES DIRECT LA MEMORIE (DMA)

Acest procedeu este folosit în cazul perifericelor rapide, adică cu o rată de transfer ridicată, de ordinul sutelor de mii de octeți pe secundă. Transferurile de date se fac direct între periferic și memorie, sub formă de blocuri, fără participarea calculatorului. Interfața cu perifericul preia întreaga comandă a transferului indicînd sensul transferului, numărul de octeți și adresa de origine din memorie.

Microprocesorul 8080 este conceput astfel încît să se poată realiza transferuri de date și prin procedeul accesului direct la memorie (*DMA*). În acest scop s-a prevăzut semnalul de tip intrare numit *HOLD*. Dacă acest semnal devine unu logic atunci microprocesorul 8080 trece într-o stare de așeptare în care cedează controlul asupra liniilor de adresă și de date, acestea trecînd în starea de impedanță ridicată. Această stare este făcută cunoscută în exterior prin trecerea semnalului *HLDA* în starea unu logic. Din acest moment interfața cu perifericul ce realizează transferul datelor prin procedeul *DMA*, poate prelua controlul asupra magistralelor de adresă și de date.

Circuitul 8257, anexă la microprocesorul 8080, este conceput anume pentru a simplifica proiectarea interfețelor cu perifericele la care se impune un transfer de date direct cu memoria. Circuitul este programabil asigurînd patru canale independente de tip *DMA* [1 ; 4].

BIBLIOGRAFIE

1. I. Dancea, ; *Microprocesoare. Arhitectură internă, programare, aplicații*, Editura Dacia, 1979
2. F. J. Hill, G. R. Peterson, *Calculatoare numerice. Hardware-structură și proiectare*, Editura tehnică, 1980.
3. C. Strugaru ; *Echipamente periferice și transmiterea datelor*, Litografia I. P. „Traian Vuia , Timișoara, 1979.
4. * * * *8080 microcomputer Systems user's manual*. 1975.

CAPITOLUL 2

REALIZAREA UNUI SISTEM CU MICROPROCESORUL 8080

VARIANTA I.

Pentru studiul funcționării microprocesorului 8080 și pentru o gamă relativ mare de aplicații, un sistem într-o configurație relativ redusă se dovedește a fi foarte util. Un astfel de sistem este bine să fie conceput în așa fel încît operatorul să poată experimenta ușor variante noi de programe și să existe un minim de facilități prin care să fie posibilă depanarea programelor sau verificarea funcționării corecte a sistemului în ansamblu sau a unor părți din sistem.

În fig. 2.1 se prezintă elementele principale ce intră în componența sistemului ce va fi analizat în cele ce urmează. Partea principală o constituie microcalculatorul organizat în jurul microprocesorului 8080. Acesta comunică în ambele sensuri cu un dispozitiv de afișare de tip display, *DAF* 1001. O mini-

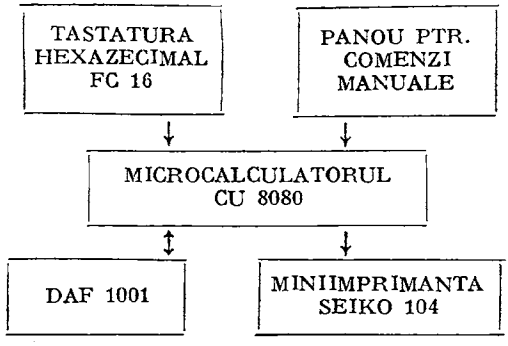


Fig. 2.1. Structura unui sistem cu 8080

imprimantă numerică tip *SEIKO* 104 este prevăzută pentru posibilitatea tipăririi pe hîrtie a unor rezultate numerice. Prin tastatura hexazecimală tip *FC*-16 model *T*03, operatorul are posibilitatea de a introduce manual în memoria *RAM* a microcalculatorului date sau programe. Partea numită „panou pentru comenzi manuale" este elementul prin care operatorul poate alege un anumit regim de funcționare, automat sau pas cu pas, poate genera o cerere de întrerupere sau poate dispune de alte facilități ce vor fi prezentate în detaliu la momentul potrivit.

Sistemul poate să fie dezvoltat în această configurație pe etape. De exemplu, în prima etapă este suficient dacă se realizează microcalculatorul și panoul pentru comenzi manuale. În etapele următoare se extinde sistemul prin atașarea celorlalte elemente în ordinea dorită. De asemenea, este posibilă extinderea în continuare a sistemului prin conectarea de noi echipamente periferice cum ar fi lectorul și perforatorul de bandă de hîrtie, caseta magnetică și altele.

2.1. CALCULUL ÎNCĂRCĂRII MAGISTRALELOR DE ADRESĂ ȘI DE DATE

În faza de proiectare a unui sistem în care se folosește microprocesorul 8080 o atenție deosebită trebuie să se acorde calculului încărcării fiecărui semnal generat de către 8080 și în mod particular a semnalelor transmise pe magistralele de adresă și de date. Acest lucru este necesar deoarece la 8080 oricare dintre semnale generate poate fi încărcat, în mod practic, cu cel mult o singură sarcină *TTL*. Din tabelul 2.1 rezultă că la microprocesorul 8080 o ieșire în starea 0 logic poate prelua un curent (I_{OL}) de numai 1,9 mA și în starea 1 logic curentul maxim generat de o ieșire (I_{OH}), poate fi de numai 150μA. Aproximativ aceleași valori sînt caracteristice și pentru celelalte circuite ce pot interveni în structura unui sistem cu 8080. Este cazul circuitelor de memorie *RAM* 2102 *A*, *EPROM* 2708 sau a circuitelor de interfață 8251, 8255, 8257 și 8259.

În sistemele în care semnalele generate de către 8080 nu sînt încărcate cu intrări *TTL*, ci numai cu intrări de circuite *MOS*, de regulă nu este necesară o amplificare în curent a acestor semnale pentru că sarcina este mică. În schimb, în sistemele în care este necesar să se prelucreze logic, prin circuite *TTL*, semnale care nu pot fi încărcate cu mai multe sarcini TTL este necesar să se prevadă circuite anume pentru amplificare în curent a acestor semnale. Este cazul, în primul rînd, a semnalelor transmise pe magistralele de adresă și de date, pentru că, în sistemele mai extinse acestea pot ajunge să fie încărcate cu un număr relativ mare de sarcini *TTL*, în deosebi magistrala de date. Pe

de altă parte trebuie avută în vedere necesitatea adaptării magistralelor, dacă lungimea lor depășește 50 cm, în scopul evitării semnalelor reflectate.

Proiectantul trebuie să țină cont pe lîngă încărcarea în curent continuu și de încărcarea capacitivă ce apare pe fiecare semnal. Datele de catalog privind timpul de propagare sînt valabile pentru sarcina capacitivă maximă admisă. Dacă această sarcină capacitivă este depășită, atunci are loc o reducere a performanțelor de catalog și pe de altă parte curentul comutat, datorită sarcinii capacitive, poate depăși valorile limită admise pentru un anumit tip de circuit.

În concluzie, pentru fiecare semnal proiectantul trebuie să stabilească :

a) încărcarea maximă ce poate să apară în curent continuu ;

b) încărcarea maximă capacitivă ;

c) dacă linia pe care se transmite un anumit semnal devine linie lungă, să adapteze linia, prin încărcarea acesteia cu o impedanță egală cu impedanța caracteristică a liniei.

În continuare se vor lua în considerare numai magistralele de adresă și de date. Dacă în urma calculelor făcute la punctul a), luînd eventual în considerare și posibilitățile de extindere în viitor ale sistemului, se ajunge la concluzia că încărcarea în curent continuu este depășită, atunci este obligatoriu să se introducă circuite de amplificare, (*BUS DRIVERS*).

Folosirea circuitului 8228 în cadrul sistemului simplifică problema în ceea ce privește magistrala de date pentru că asigură pentru liniile de date în starea zero un curent I_{OL} de la 10 mA (Tabelul 2.1). În această situație magistrala de date poate fi încărcată cu pînă la 6 sarcini *TTL*. În plus circuitul 8228 asigură și caracterul bidirecțional al magistralei de date.

Tabelul 2.1

Încărcarea admisă a semnalelor de intrare și de ieșire

Circuitul	I_{OL} [mA]	I_{OH} [μA]	I_{IL} [μA]	I_{IH} [μA]
8080 A	1,9	−150	−100	10
8251 A	2,2	−400	10	10
8255	1,6	−100	10	10
8257	1,6	−150	10	10
8259	2,0	−400	10	10
8228	10,0	-10^3	250	250
8212	15,0	-10^3	−250	10
8216	50,0	-10^4	−250	10
2102 A	2,1	−100	10	10
2708	1,6	−100	10	10
S N 74126	24,0	$-5{,}2\ 10^3$	$-1{,}6\ 10^3$	40
SN 7438	48,0	−250	$-1{,}6\ 10^3$	40
8T97	48,0	$-5{,}2\ 10^3$	−400	40

În familia circuitelor *INTEL* 8080 sînt prevăzute și circuite anume destinate amplificării semnalelor transmise pe magistrale. Este cazul circuitelor 8216 și 8226, circuite bidirecționale pe patru biți, care asigură un curent de sarcină la ieșirea în starea zero logic de 25 mA și în starea unu logic de —10 mA. Pentru nivelul de 0,5—0,6 volt la ieșirea în starea zero logic, curentul absorbit poate fi de 50 mA. De remarcat faptul că aceste circuite au un consum mic pe intrări, 0,25 mA în starea zero logic și 10 μA în starea unu logic, asigurînd pe ieșirile (DO) ce se leagă la magistrala de date a microprocesorului niveluri de minim 3,65 volt în starea unu logic. Acest lucru este important întrucît nivelul minim de intrare în starea unu logic cerut de 8080 este de 3,3 volt.

Un alt circuit ce poate fi folosit în scopuri de amplificare este circuitul 8212. Pentru magistrala de date, cu două circuite 8212 legate în opoziție se asigură caracterul bidirecțional. Amplificarea lor în curent nu este mare, $I_{OL}=15$ mA (Tabelul 2.1), au însă un consum mic pe intrări ($I_{IL}=-0{,}25$ mA) și o valoare ridicată a nivelului pentru ieșirile în starea unu logic ($V_{OH}=3{,}65$ volt minim), fiind din acest motiv direct compatibile cu intrările lui 8080. Utilizarea lui 8212 ca amplificator de magistrală nu valorifică toate posibilitățile acestui circuit și comparativ cu folosirea lui 8216 sau 8226 constituie o soluție mai scumpă și cu performanțe mai reduse.

O altă posibilitate economică de amplificare a semnalelor transmise pe magistrale, folosind porți cu trei stări (SN74126), este prezentată în fig. 2.2. Expresiile logice pentru semnalele de validare a porților sînt următoarele

$$V_A = \overline{HLDA}$$

$$V_{DI} = DBIN.\overline{HLDA}$$

$$V_{DO} = \overline{DBIN}.\overline{HLDA}$$

sau :

$$V_{OD} = (\overline{WR}).\ \overline{HLDA}.$$

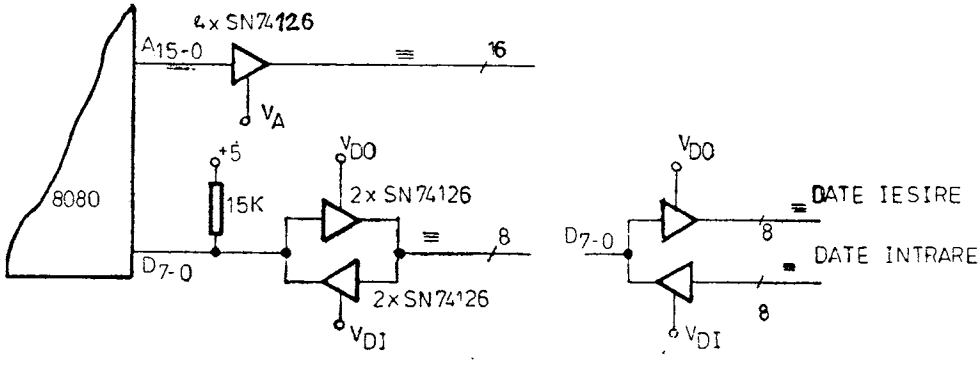


Fig. 2.2. Amplificarea semnalelor pe magistralele de adresă și date

Se observă că dacă 8080 se găsește în starea *HOLD* atunci $HLDA=1$ și în acest caz toate semnalele de validare se găsesc în starea zero logic, ceea ce provoacă trecerea ieșirilor în starea de impedanță ridicată. Dacă 8080 nu se găsește în starea *HOLD* atunci $HLDA=0$. Cînd 8080 citește starea magistralei de date, activează semnalul ($DBIN=1$) și în acest caz $V_{DI}=1$ și $V_{DO}=0$. Rezistoarele de 15 Kohmi, legate la +5 volt, au rolul de a fixa pe liniile de date, ce se găsesc în starea unu logic, un nivel de tensiune V_{IH} mai mare de 3,3 volt, cît este minim necesar pentru 8080. Rezistoare de o valoare mai mică, de exemplu 5 Kohmi, produc o încărcare suplimentară, de cca 1 mA în starea 0 logic la ieșire, care se adaugă la sarcina de 1,6 mA, formată de către poarta de ieșire, depășind astfel limita de 1,9 mA.

Magistrala de date poate fi ramificată în două magistrale separate, una pentru ieșiri de date și a doua pentru intrări de date, așa cum se indică în partea din dreapta în fig. 2.1. Magistrala pentru ieșiri de date nu trebuie prevăzută obligatoriu cu porți 3-stări întrucît va fi legată numai la intrări de circuite logice.

În fig. 2.3 se ilustrează utilizarea principiului ramificării magistralelor de adresă și de date. Magistralele de ieșire nu sînt prevăzute cu porți 3-stări, permițînd o încărcare la ieșire cu 30 sarcini *TTL*.

Dubla inversare pentru magistralele de ieșire se face cu scopul de a menține aceleași convenții pe ambele magistrale. Magistrala de adresă de tipul 3 stări poate fi poziționată, atunci cînd 8080 se găsește în starea *HOLD*, prin circuite cu ieșiri 3 stări sau prin circuite cu colector în gol. Datele de intrare vor fi preluate de 8080 prin magistrala de date de tipul 3 stări. Această magistrală va putea fi poziționată prin circuite de tipul 3 stări sau prin circuite cu colector în gol. În cazul folosirii circuitelor cu colector în gol se va avea în

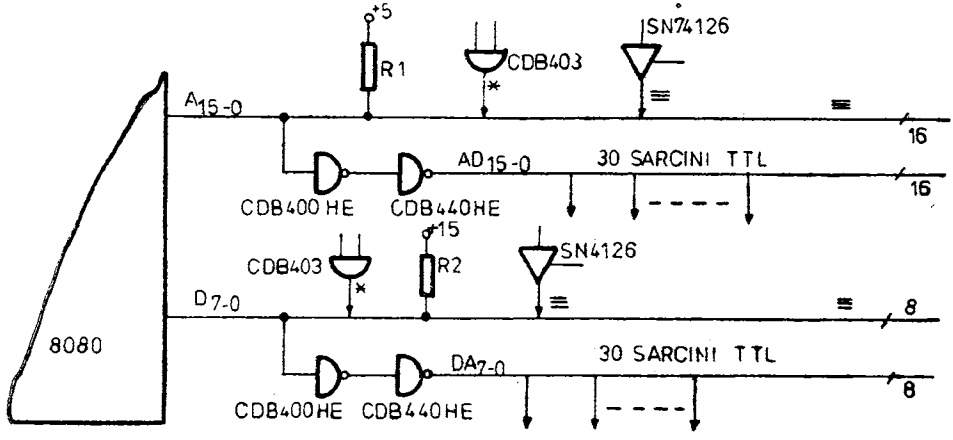


Fig. 2.3. Amplificarea semnalelor pe magistralele de ieșire

vedere că timpul de comutare inversă este mai mare decît la circuitul cu 3 stări. Acest fapt prezintă importanță mai ales dacă microprocesorul funcționează la frecvența de tact maximă.

În cazul că se adoptă soluția cu magistrale de daté separate pentru intrări și ieșiri, apare o problemă la conectarea circuitelor de interfață de tipul 8251, 8255 ș.a., la care magistrala de date este bidirecțională și suportă aproximativ aceleași încărcări ca și magistrala de date a lui 8080. Datorită consumului redus pe intrări astfel de circuite se pot conecta pe liniile D_{7-0} înainte de amplificatoarele de magistrală. În acest caz însă este necesar ca semnalul V_{DI} să fie astfel condiționat logic încît să nu devină activ atunci cînd 8080 preia date de intrare de la aceste circuite. În caz contrar ar rezulta că sînt activate simultan, pe fiecare linie de date, două porți de tipul 3 stări, una fiind cea comandată de semnalul V_{DI} și cealaltă aparținînd tamponului de ieșire al circuitului 8251 sau 8255. În cazul schemei din fig. 2.3 lucrurile sînt mai simple deoarece astfel de circuite se pot conecta direct la magistrala D_{7-0} de tipul 3 sătri. Pe aceeași magistrală se pot conecta și alte circuite cu consum foarte mic pe intrări (circuitele pentru memoriile *RAM*, *ROM*), avînd totodată în vedere și cele menționate anterior referitor la încărcarea în curent continuu și sarcina capacitivă.

În sistemele mai complexe, în care lungimea fizică a magistralelor de date și de adresă depășește 50 cm, acestea capătă caracterul unor linii lungi și pentru a evita influența semnalelor reflectate este necesar să se asigure o sarcină la capătul liniei, egală cu impedanța caracteristică Z_0, a liniei (fig. 2.4).

Pentru conductorul bifilar răsucit, cu pas de răsucire egal cu un cm, se poate considera ca valoare tipică pentru Z_0 cca 120—150 Ω. Rezistoarele $R1$ și $R2$ în paralel trebuie să dea o valoare egală cu valoarea lui Z_0. În starea 0 logic pe linie. curentul I_{LL} este practic determinat de rezistorul $R1$ și va încărca corespunzător ieșirea emițătorului. În starea unu logic se urmărește să se obțină pe linie un nivel de tensiune cît mai ridicat, ceea ce ar însemna o valoare relativ mică pentru $R1$ în raport cu $R2$. O valoare mică penrru $R1$ conduce la o încărcare mare a emițătorului în starea zero logic la ieșire. Din aceste con-

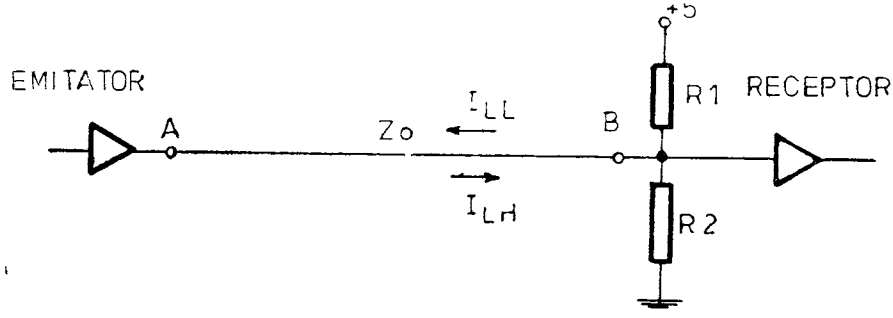


Fig. 2.4. Adaptarea liniilor lungi

siderente se calculează adaptorul (numit uneori și terminator) ținînd cont și de tipul circuitului emițător. Pentru $R1=180$ Ohmi și $R2=390$ Ohmi linia va fi încărcată cu o impedanță de sarcină egală cu 120 Ω și în punctul E adaptorul fixează nivelul $V_E=3{,}4$ volt.

În starea 0 logic curentul I_{LL} absorbit de emițător se compune din curentul generat de către receptor și curentul ce trece prin $R1$, considerînd $V_E=0{,}4$ volt. În acest caz $I_{R1}=25$ mA ceea ce reprezintă aproximativ 15 sarcini TTL pentru emițător.

2.2. ORGANIZAREA MICROCALCULATORULUI

În fig. 2.5 se prezintă cum anume sînt conectate între ele circuitele ce intră în componența microcalculatorului realizat în jurul microprocesorului 8080. Sînt indicate, de asemenea, o parte dintre legăturile ce trebuie să existe cu celelalte elemente date în fig. 2.1.

S-a prevăzut, pentru început, o memorie *RAM* de 1 Koctet realizată cu capsule de memorie *RAM* tip 2102 A și o memorie *ROM* de 1 Koctet realizată cu o capsulă de memorie *EPROM* tip 2708.

Octetul de stare furnizat de către 8080 la începutul fiecărui ciclu $M1$ este memorat într-un registru format din două capsule *CDB* 495, fiecare bit fiind vizualizat printr-un *LED*.

Semnalele de comandă generate de către 8080 și 8224 sînt amplificate prin porți rapide (*CDB* 411 *HE*).

Starea fiecărei linii din magistralele de adresă și de date este vizualizată printr-un *LED*, aprins cînd linia respectivă se găsește în starea unu logic.

Se mai indică în fig. 2.5 că magistralele de adresă și de date mai pot fi poziționate prin intermediul panoului, operație ce are loc numai atunci cînd 8080 se găsește în starea *HOLD*. Tot în această stare este posibil a poziționa magistrala de date de la tastatura hexazecimală. Magistrala de date mai este conectată și cu circuitele de interfață la DAF 1001 și cu miniimprimante, neindicate însă în fig. 2.5.

2.3. SELECTAREA BLOCURILOR DE MEMORIE

Spațiul de memorie adresabil cu cele 16 ranguri de adresă (A_{15-0}) este cuprins între 0 și 65 535 locații. Cum în majoritatea aplicațiilor este suficient un spațiu de memorare mult mai redus, s-a limitat capacitatea maximă totală

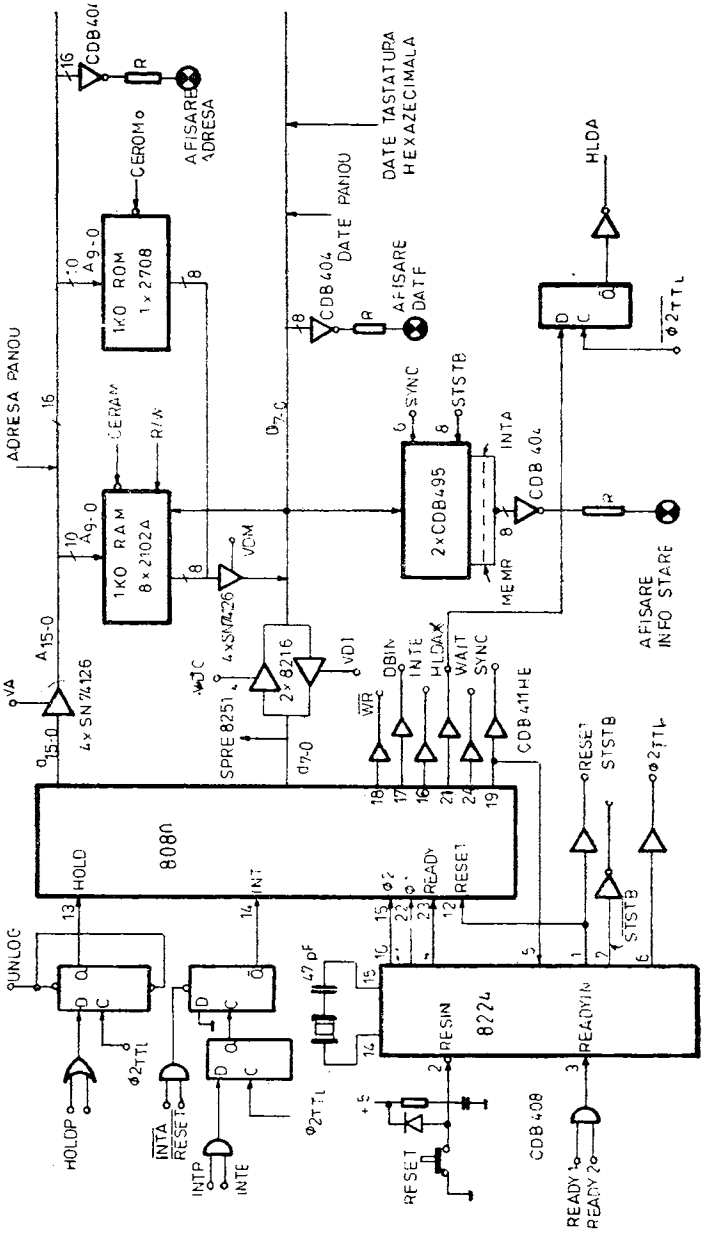


Fig. 2.5. Organizarea microcalculatorului bazat pe 8080

a blocurilor de memorie la jumătate și anume între adresa 0 și 32 767 adică la 32 Kocteți. În acest caz orice adresă cuprinsă între aceste limite va avea bitul cel mai semnificativ, A_{15}, egal cu zero. Toate adresele peste această valoare vor avea bitul A_{15} egal cu 1 și ele vor servi la selecția unui echipament periferic. În acest fel operațiile de intrare și ieșire se vor realiza prin *I/I* în spațiul memoriei.

Revenind la blocurile de memorie indicate în fig. 2.5, memoria *RAM* ocupă zona cuprinsă între adresele 0 la 1023 și memoria *ROM* zona cuprinsă între adresele 1024 la 2047. Repartizînd astfel blocurile de memorie în spațiul adreselor, rezultă că în *RAM*, începînd de la adresa zero, trebuie să fie înregistrat programul ce urmează a fi executat. Acest lucru este necesar deoarece la pornire, sub acțiunea semnalului *RESET* microprocesorul 8080 înscrie în numărătorul de program (*PC*) adresa zero și prin urmare prima instrucție ce va fi executată va fi cea înregistrată începînd de la adresa zero.

Este posibil totuși să se execute programe înregistrate în *ROM*, dacă la adresa zero din *RAM* se introduce manual, înainte de lansarea în execuție automată, o instrucție de salt necondiționat (*JMP*) la prima instrucție a programului existent în *ROM*. Există și posibilitatea de a schimba distribuția celor două blocuri de memorie în spațiul adreselor și anume atribuind adresele 0 la 1023 memoriei *ROM* și adresele 1024 la 2047 memoriei *RAM*. În acest caz este necesar ca în *ROM* începînd de la adresa zero să fie deja înregistrat un anume program. Înregistrarea unui program în memoria 2708 se face cu un echipament anume destinat acestui scop care creează așa numitul „regim de programare" pentru capsula 2708. În acest scop poate fi folosită chiar schema din fig. 2.5. așa cum se va arăta mai tîrziu. La început, presupunînd că nu există încă posibilitatea de a programa memoria 2708, este indicat a plasa memoria *RAM* începînd de la adresa zero. La selecția blocurilor de memorie s-a ținut cont de posibilitatea de extindere pînă la 4 Kocteți, selecția blocurilor de memorie făcîndu-se cu rangurile de adresă A_{11}, A_{10} și A_{15} în felul următor :

$$CERAM=\overline{\overline{A15}.\overline{A11}.\overline{A10}}$$

$$CEROM_0=\overline{\overline{A15}.\overline{A11}.A10}$$

$$CEROM_1=\overline{\overline{A15}.A11.\overline{A10}}$$

$$CEROM_2=\overline{\overline{A15}.A11.A10}$$

Pentru a trece blocul de memorie ROM_0 pe adresele 0 la 1023 se poate folosi soluția dată în fig. 2.6. Dacă comutatorul se găsește în poziția desenată în figură atunci memoria *RAM* ocupă adresele 0 la 1023 și memoria *ROM* ocupă adresele 1024 la 2047. Prin trecerea comutatorului glisant spre stînga se inversează pozițiile ocupate de către cele două blocuri. Această schimbare a poziției comutatorului glisant se poate face și atunci cînd schema se găsește

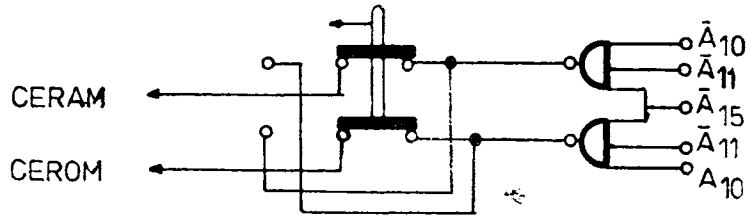


Fig. 2.6. Modificarea alocării spațiilor de adresă pentru memoriile ROM și RAM

sub tensiune cu condiția ca microprocesorul să se găsească în starea *HOLD*. Evident că același rezultat se obține intervenind în cablaj și schimbînd cele două semnale *CERAM* și $CEROM_0$ între ele.

2.4. GENERAREA SEMNALELOR DE COMANDĂ

În organizarea microcalculatorului din fig. 2.5 în mod intenționat nu s-a prevăzut circuitul 8228 (System Controller), pentru a arăta că funcțiile atribuite semnalelor generate de către acest circuit pot fi de fapt preluate de către semnale cu destinație similară, generate însă prin scheme logice combinaționale. Această substituire a circuitului 8228 se face pe seama unui consum mai mare de capsule. (Funcția de amplificare a semnalelor transmise pe magistrala de date s-a discutat în Cap. 2.1).

Expresiile logice pentru semnalele *VA* (validare adresă), *VDO* (validare date ieșire), *VDI* (validare date intrare), *VDM* (validare date din memorie) și *R/W* (citire sau scriere în memoria *RAM*) ce intervin în fig. 2.5 vor fi analizate în continuare.

Semnalul *VA* are ecuația :

$$VA = \overline{HLDA}.$$

Atunci cînd 8080 se găsește în starea *HOLD*, adică $HLDA = 1$, rezultă că $VA = 0$.

Semnalele *VDO* și *VDI* au fost trecute sub această formă în fig. 2.5 mai mult sub formă simbolică. Ca amplificatoare pe magistrala de date se indică folosirea circuitelor 8216 la care validarea datelor de intrare și ieșire se comandă cu semnalele $\overline{CS}$ și $\overline{DIEN}$ generate astfel :

$$\overline{CS}_{8216} = HLDA\ ;$$
$$\overline{DIEN} = DBIN.$$

În starea *HOLD* circuitul 8216 pune magistrala de date în starea de impedanță ridicată deoarece semnalul $\overline{CS}_{8216}$ este activ în starea zero. Cînd 8080 nu se găsește în starea *HOLD*, atunci sensul transferului pe magistrala de date este determinat de semnalul $\overline{DIEN}$. Dacă $\overline{DIEN}=1$, deci în timpul cît *DBIN* este activ, atunci sensul de transfer pe magistrala de date este din exterior spre 8080 și dacă $\overline{DIEN}=0$, sensul de transfer este opus.

Semnalul *VDM* are ecuația :

$$VDM=MEMR.DBIN.\overline{A15}+EXAM,$$

unde termenul *EXAM* este generat de către panoul de comandă și atîta timp cît acesta este inactiv avem $EXAM=0$.

Semnalul *R/W* are ecuația :

$$R/W=\overline{(WR).A15.MEM.MEMT},$$

unde semnalele $\overline{MEM}$ și $\overline{MEMT}$ sînt generate de către panoul de comandă și respectiv tastatura hexazecimală și atîta timp cît acestea sînt inactive starea semnalelor respective este 1 logic.

2.5. PANOUL DE COMANDĂ

Panoul de comandă este conceput astfel încît să ofere operatorului următoarele posibilități :

1 — Să aducă microprocesorul în starea *HOLD*.

2 — Să realizeze operații de scriere în memoria *RAM*, începînd de la o adresă stabilită de către operator.

3 — Să realizeze operații de vizualizare a octeților înscriși în memoria *RAM*, începînd de la o adresă stabilită de către operator.

4 — Să lanseze o cerere de întrerupere către 8080.

5 — Să aleagă un regim de lucru pentru 8080 din trei regimuri posibile și anume : regimul automat, regimul instrucție cu instrucție și regimul ciclu cu ciclu.

Pentru realizarea cerințelor enumerate mai sus pe panou sînt prevăzute următoarele elemente :

a) $CHEI_{7-0}$ — un grup de 8 comutatoare bipoziționale prin care operatorul formează un octet.

b) *BLO* — un buton cu apăsare/revenire prin acționarea căruia în registrul de adrese al panoului, *RADRP*, se încarcă octetul mai puțin semnificativ al adresei (A_{7-0}), format prin poziționarea corespunzătoare a grupului de $CHEI_{7-0}$.

c) *BHI* — un buton cu apăsare/revenire prin acționarea căruia în RADRP se încarcă octetul mai semnificativ al adresei (A_{15-8}), format prin poziționarea corespunzătoare a grupului de $CHEI_{7-0}$.

d) *BMEM* — un buton cu apăsare/revenire prin acționarea căruia octetul format prin poziționarea corespunzătoare a grupului de $CHEI_{7-0}$, se înscrie în *RAM* la adresa existentă în *RADRP*. La revenirea butonului *BMEM* în starea inițială, conținutul din *RADRP* crește cu o unitate, formîndu-se astfel adresa următoarei locații din memoria *RAM*.

e) *BEXAM* — un buton cu apăsare/revenire prin acționarea căruia conținutul locației din memoria *RAM*, de la adresa existentă în acel moment în *RADPR*, apare pe liniile D_{7-0}. La revenirea butonului *BEXAM* în starea inițială conținutul din *RADRP* crește cu o unitate.

f) *CHOLD* — un comutator bipozițional. În poziția „sus" generează semnalul *HOLDP*=1 care după sincronizare devine semnalul de intrare *HOLD* pentru 8080 (fig. 2.5).

g) *CINT* — un comutator bipozițional. În poziția „sus" generează semnalul de întrerupere de tip panou (*INTP*).

h) *BRESET* — un buton cu apăsare/revenire prin acționarea căruia circuitul 8224 generează semnalul *RESET* la 8080.

i) *STEP, STEPM, STEPI* — trei comutatoare bipozitionale cu ajutorul cărora se pot realiza cele trei regimuri diferite de funcționare pentru 8080. Rolul fiecăruia se va analiza ulterior.

Pe panou sînt montate *LED*-urile pentru vizualizarea stării liniilor de adresă, de date și a informației de stare (fig. 2.5).

2.5.1. ÎNCĂRCAREA MEMORIEI RAM DE LA PANOU

În acest scop microprocesorul trebuie adus în starea *HOLD* prin acționarea comutatorului *CHOLD*. În momentul în care 8080 cedează controlul asupra liniilor A_{15-0} și D_{7-0} atunci $HLDA=1$. Registrul de adrese al panoului, *RADRP*, (fig. 2.7) este format din două capsule *CDB* 495, în jumătatea mai semnificativă, și din două numărătoare cu încărcare *SN* 74163 (în locul acestora se pot folosi numărătoarele *CDB* 4193 legate corespunzător) în jumătatea mai puțin semnificativă.

Semnalele *VAP* (validare adresă de la panou) și *VDP* (validare date de la panou) din fig. 2.7 au următoarele expresii :

$$VAP=HOLDP\cdot HLDA\ ;$$

$$VDP=HOLDP\cdot HLDA\cdot MEM+INTA\cdot DBIN.$$

La acționarea butonului *BLO* (fig. 2.8) se creează pentru ambele numărătoare *SN* 74163 regimul de încărcare deoarece $LOAD=\overline{LO}=0$. Semnalul *LO* declanșează monostabilul *MON* 1 și acesta formează semnalul *CLOCK* care

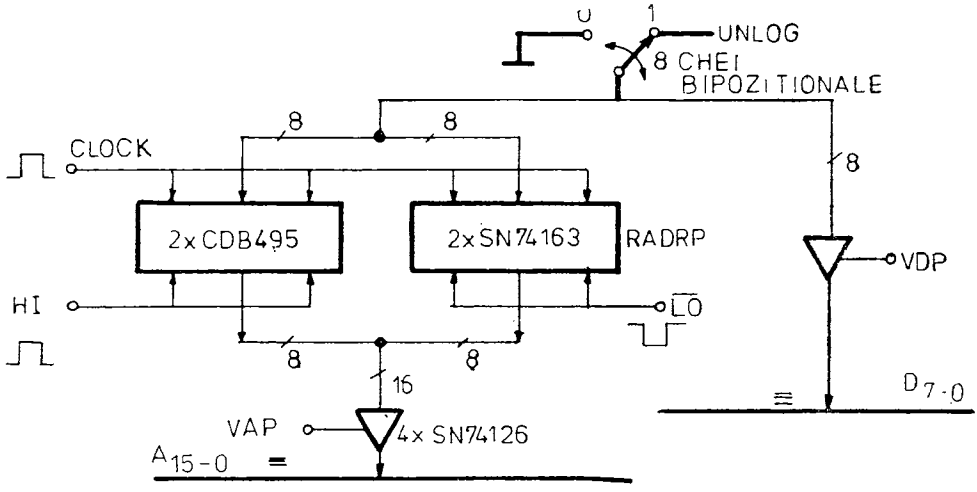


Fig. 2.7. Registrul de adrese al panoului

încarcă starea celor 8 chei în jumătatea mai puțin semnificativă din *RADRP*. După schimbarea poziției cheilor se acționează butonul *BHI* și în mod similar se încarcă octetul mai semnificativ al adresei în registrele *CDB* 495. Se poziționează cele 8 chei pentru a forma un octet de date. După această operație, dacă se acționează butonul *BMEM*, atunci pe liniile D_{7-0} se poziționează octetul de date fixat prin chei, deoarece $VDP=1$ datorită termenului $HOLDP \cdot HLDA \cdot MEM=1$ iar semnalul R/W devine zero întrucît $\overline{MEM}=0$. În memoria RAM, la adresa încărcată în *RADRP*, se va înscrie octetul existent pe liniile D_{7-0} care poate fi urmărit la cele 8 *LED*-uri ce vizualizează liniile de date. La eliberarea butonului *BMEM*, liniile de date, D_{7-0}, revin toate în starea unu logic și monostabilul *MON*2 generează semnalul *CLOCK* care mărește cu o unitate conținutul celor două numărătoare *SN* 74163 legate în cascadă. În felul acesta se formează adresa locației următoare din *RAM*. Pentru a pune numărătoarele *SN* 74163 în regim de numărare intrările *ENP* și *ENT* sînt legate la bistabilul *ENPT*, comandat așa cum se arată în fig. 2.8.

Pentru vizualizarea conținutului unei locații din RAM se acționează butonul *BEXAM*. Dacă $EXAM=1$ atunci $VDM=1$ și conținutul locației din *RAM*, de la adresa existentă în *RADRP*, apare pe liniile de date D_{7-0}. La eliberarea butonului, *RADRP* avansează cu o unitate. Deoarece în *RADRP* s-au folosit numai două numărătoare în cascadă, adresa avansează automat, la acționarea butoanelor *BMEM* sau *BEXAM*, pînă la valoarea 255 după care este necesară o nouă operație manuală de încărcare a registrelor *CDB* 495 cu octetul superior al adresei, mărit cu o unitate.

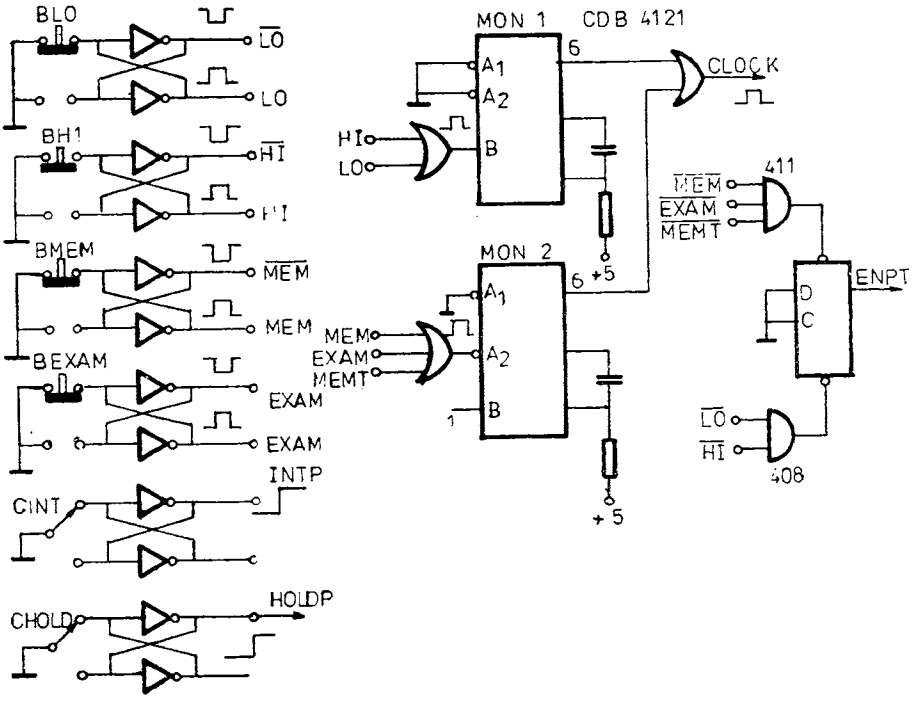


Fig. 2.8. Comenzi generate de la panou

2.5.2. REGIMURI DE FUNCȚIONARE

De la panou se pot stabili cu ajutorul schemei din fig. 2.9 trei regimuri de funcționare pentru 8080

Regimul automat. În regimul automat 8080 execută programul din *RAM* la frecvența dată de generatorul de tact 8224. După alimentarea cu tensiune a sistemului se acționează butonul *RESET* (fig. 2.5) avînd poziționat comutatorul *CHOLDP* astfel ca *HOLDP* să fie unu logic. Microprocesorul va intra în starea *HOLD*. În această stare, se poate încărca de la panou în *RAM*, începînd de la adresa zero, programul dorit. După încărcare se repoziționează adresa zero în *RADRP* și prin acționarea repetată a butonului *BEXAM* se poate vizualiza octet după octet, informația înscrisă în *RAM*. De fapt, un program poate fi înregistrat în orice zonă din *RAM* cu condiția ca la adresa zero să se memoreze o instrucție de salt necondiționat la prima instrucție a programului.

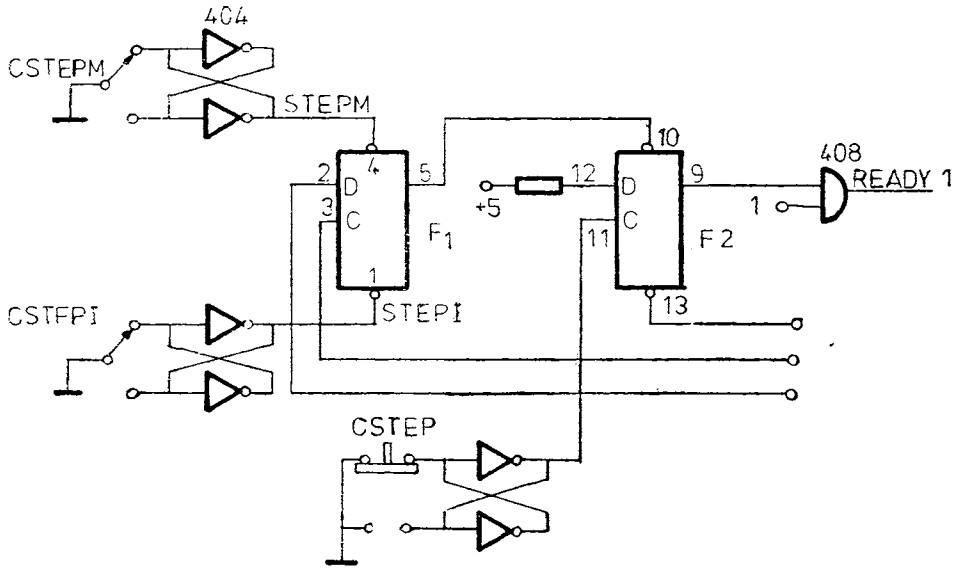


Fig. 2.9. Selecția regimului de funcționare

Se poziționează comutatoarele astfel ca $STEPM=1$ și $STEPI=0$. Se acționează butonul *RESET* și în timp ce acesta este acționat se comută *CHOLD* pe poziția în care $HOLDP=0$, după care se eliberează butonul *RESET*. Microprocesorul va trece la execuția automată a programului memorat începînd de la adresa zero.

Regimul instrucție cu instrucție. Acest regim se realizează cu $STEPM=1$ și $STEPI=1$. În timp ce *RESET* este acționat se comută comutatorul *CHOLD* pentru a scoate pe 8080 din starea *HOLD*. Se eliberează butonul *RESET* și 8080 intră în ciclul $M1$ de extragere a instrucției. În informația de stare bitul $M1$, care apare pe linia $D5$, va fi unu și *STSTB* va poziționa bistabilul $F1$ în starea unu. Deoarece $WAIT=0$, bistabilul $F2$ va fi în starea zero și întrucît $READY\ 1=0$, 8080 va trece în starea de așteptare *TW*. În această stare avem $WAIT=1$. Adresa va fi afișată în binar de către *LED*-urile de adresă și conținutul adresei respective din *RAM* va fi afișat de către *LED*-urile de date. Dacă se acționează butonul *CSTEP* atunci bistabilul $F2$ trece în starea unu și 8080 părăsește starea *TW*. În ciclul următor, $M2$, în informația de stare bitul $M1$ va fi pe zero și bistabilul $F1$ va trece în starea zero. În acest caz $F2$ va trece în starea unu deoarece intrările de comandă asincrone se găsesc ambele pe zero și rezultă că $READY1=1$. Din acest motiv 8080 nu se mai oprește în ciclul $M2$. La fel se petrec lucrurile în ciclul $M3$ și în ciclurile

M4 și *M5* dacă ele există. Rezultă că în acest regim 8080 se va opri în starea *TW* din cadrul fiecărui ciclu *M1*, de extragere a primului octet dintr-o instrucție.

Regimul ciclu cu ciclu. Acest regim se realizează cu $STEPM=0$ și $STEPI=1$. Se procedează ca mai sus pentru a lansa microprocesorul în execuția programului. Deoarece $STEPM=0$ bistabilul $F2$ va fi tot timpul în stare unu și în acest caz $F2$ va trece sub acțiunea lui *WAIT* în starea zero în fiecare ciclu de mașină. Prin acționarea butonului *CSTEP* bistabilul $F2$ trece în starea unu și 8080 poate părăsi starea *TW*.

Acest regim este foarte util în faza de punere la punct a sistemului precum și în urmărirea modului în care se realizează operațiile cu echipamentele de intrare-ieșire conectate la sistem.

2.5.3. ÎNTRERUPERI GENERATE DE LA PANOU

De la panou există posibilitatea lansării unei cereri de întrerupere prin acționarea comutatorului *CINT*. Semnalul INTP, dacă întreruperile sînt validate prin $INTE=1$, conduce la trecerea în starea unu logic a semnalului de intrare *INT*. În secvența de tratare a cererii de întrerupere, termenul *INTA* . *DBIN* din expresia semnalului *VDP*, va poziționa pe magistrala D_{7-0} octetul format de către operator la panou prin cheile de date ($CHEI_{7-0}$). Octetul respectiv va fi preluat de către 8080 ca o instrucție *RSTn*, saltul având loc la adresa $8 \times n$.

Semnalul *INTP* poate fi tratat în schema din fig. 2.5 și ca un semnal de acceptare al unei cereri de întrerupere generată de către un alt echipament, din cadrul sistemului sau din afara lui, caz în care operatorul prin poziționarea corespunzătoare a cheilor de date existente pe panou, poate alege subrutina dorită pentru tratarea acelei întreruperi.

2.5.4. TASTATURA HEXAZECIMALĂ

Pentru comoditatea introducerii datelor în memoria RAM în timp ce 8080 se găsește în starea *HOLD* s-a atașat la sistem tastatura hexazecimală tip *FC* 16 Model *T*03 conform schemei de interfață dată în fig. 2.10. La apăsarea pe prima tastă, codul tastei, de pe liniile RCT_{1-4} care vin de la tastatură, se încarcă în registrul *CDB* 495 ; pe linia *CIMP* apare semnalul de validare al codului și acesta comandă trecerea bistabilului *B* în starea unu. La apăsarea tastei următoare conținutul registrului *CDB* 495 este poziționat pe liniile D_{7-4} și codul ultimei taste pe liniile D_{3-0}. Semnalul $\overline{MEMT}$ ce intervine în expresia lui *R/W* provoacă operația de scriere în *RAM* și semnalul *MEMT* comandă

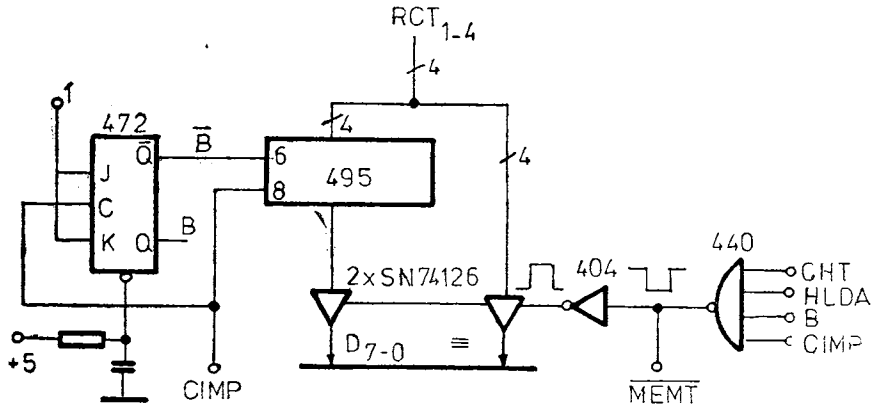


Fig. 2.10. Interfața cu testatura hexazecimală

avansul cu o unitate a conținutului registrului *RADRP*. Semnalul *CHT* provine de la un comutator bipoziţional, ce poate fi montat de asemenea pe panou. Dacă CHT=0 atunci se elimină influența tastaturii asupra liniilor de date prin acționarea accidentală a tastelor în timpul funcționării sistemului.

Dacă în timpul introducerii datelor de la tastatura hexazecimală, un octet s-a memorat prin acționarea din greșeală a unei alte taste, atunci pentru octetul respectiv operația se reia cu încărcarea în *RADRP* a adresei corecte, deoarece în *RADRP* s-a format deja adresa următorului octet. De reținut și faptul că *RADRP* funcționează ca numărător cu încărcare numai pentru octetul mai puțin semnificativ din adresă. Din acest motiv cînd se ajunge cu octetul inferior din adresă la valoarea *FF*, avansul cu o unitate conduce la valoarea 00 iar transportul în rangul următor se pierde. Operatorul va fi obligat să încarce în *RADRP*, în jumătatea superioară, o valoare egală cu a octetului anterior plus o unitate. Această operație se elimină dacă *RADRP* este realizat în întregime cu numărătoare sincrone programabile de tipul *SN* 74163 sau *SN* 74193.

În fig. 2.11 se arată un mod posibil de amplasare a comutatoarelor, butoanelor și *LED*-urilor pe partea frontală a panoului de comandă. O vedere a acestei plăci se prezintă în fotografia 2.1. Placa frontală din aluminiu îndeplinește numai funcția unei măști, toate elementele fiind rigid fixate pe placa din spate ce conține componentele electronice specifice panoului.

În fotografia 2.2 se prezintă o fotografie a plăcii pe care este realizat microcalculatorul cu schema dată în fig. 2.5. Placa folosită este de tipul *CIW*-02 proiectată și realizată la *ITC*-Timișoara. Componentele se implantează în socluri cu pini de tipul Wire-wrapping.

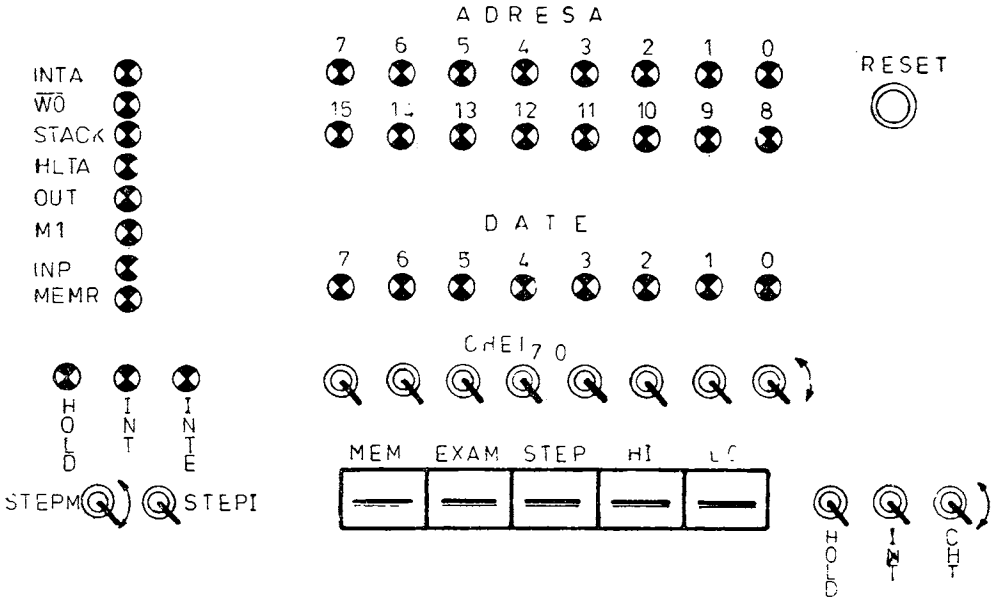


Fig. 2.11. Amplasarea elementelor pe panoul de comandă

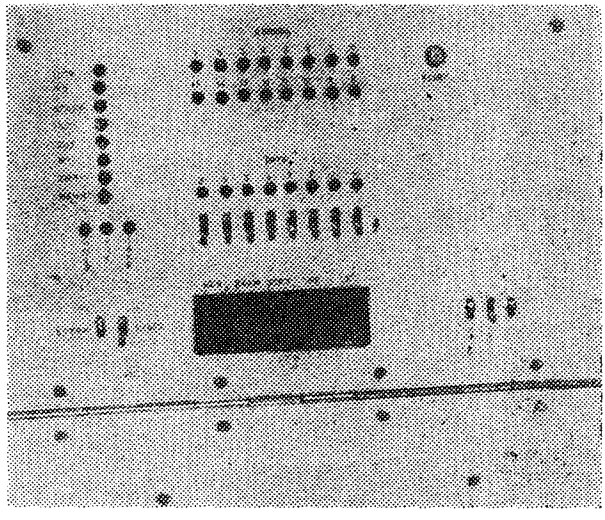


Foto 2.1. Panoul de comandă al microcalculatorului

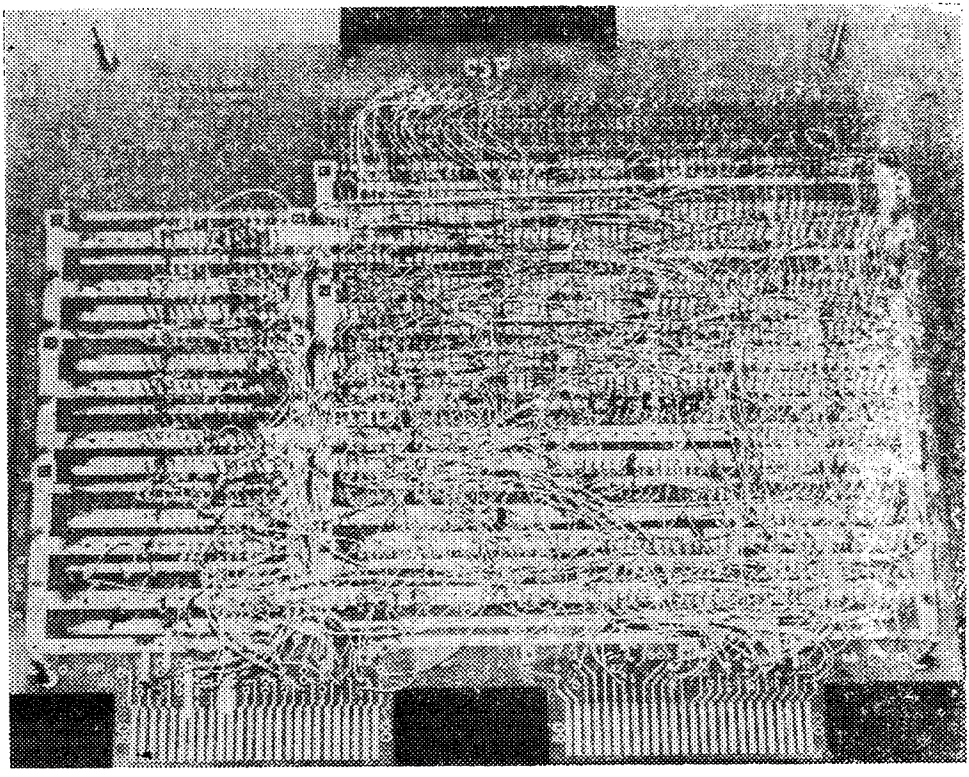


Foto 2.2. Fața cablată a plăcii microcalculatorului

2.6. MODUL DE REALIZARE A OPERAȚIILOR DE INTRARE-IEȘIRE

Microprocesorul 8080 pentru realizarea operațiilor de intrare-ieșire dispune numai de instrucțiile *IN* și *OUT*, prin care se pot selecta pînă la maximum 256 de echipamente periferice, respectiv porturi de intrare-ieșire. Pentru a oferi programatorului o flexibilitate mai mare în efectuarea operațiilor de intrare-ieșire, aceste operații vor fi realizate prin *I/I* în spațiul memoriei.

Aceasta înseamnă că oricare din instrucțiile lui 8080 cu referire la memorie se poate folosi și pentru efectuarea operațiilor de intrare-ieșire, cu condiția ca adresa implicată în procesul de adresare să fie în afara limitelor atribuite pentru zona de memorare propriu-zisă. În cap. 2.3 s-a convenit asupra acestui

lucru, în sensul că adresele care au bitul A_{15} egal cu unu nu vor fi interpretate ca adrese ale unor locații din zona de memorare ci ca adrese pentru selecția porturilor de intrare-ieșire. De exemplu, adresa 8001, scrisă în hexazecimal, în care $A_{15}=A_0=1$, va servi pentru selecția unui registru tampon din interfața cu miniimprimanta. Încărcarea acestui registru tampon cu octetul existent în registrul B al microprocesorului se va putea face cu instrucția *MOV M, B* dacă în timpul execuției acestei instrucții în registrele H și L se găsește adresa 8001. În ciclul M2 al instrucției *MOV M, B* pe magistrala de adresă. A_{15-0}, se poziționează conținutul registrelor H și L și pe magistrala de date. D_{7-0}, conținutul registrului B [1]. În timpul stării $T3$ din cadrul ciclului $M2$ sub comanda semnalului $\overline{WR}$ are loc încărcarea registrului tampon. În acest timp nici unul dintre blocurile de memorie, nu va fi selectat, întrucît $A_{15}=1$.

Selecția porturilor și realizarea efectivă a operației de încărcare, respectiv de citire, nu este discutată în detaliu aici, deoarece acest lucru depinde de tipul circuitelor folosite ca porturi de intrare-ieșire. În tabelul 2.2. se prezintă repartizarea adreselor de selecție pentru porturile de intrare și ieșire, ce apar în cadrul circuitelor de interfață, tratate ulterior în cadrul acestui capitol. De remarcat faptul că o aceeași adresă a fost atribuită atît pentru un port de intrare, la care 8080 realizează o operație de citire, cît și pentru un port de ieșire, la care 8080 realizează o operație de încărcare, adică scriere. Acest lucru este posibil dacă semnalul de selecție a porturilor de intrare este condiționat cu *MEMR*, din octetul de stare, și semnalul de selecție a porturilor de ieșire cu semnalul $\overline{W_0}$ din informația de stare.

În tabelul 2.2. sînt indicate, cu polaritatea corespunzătoare, semnalele de comandă *SCR* și *CIT*. Cele notate cu mnemonicul *SCR* sînt folosite în operații de scriere la nivelul porturilor de ieșire și cele notate cu *CIT* în operații de citire la nivelul porturilor de intrare.

Din paragrafele următoare ale acestui capitol va rezulta în detaliu rolul fiecărui semnal în parte; în coloana cu explicații sînt făcute unele precizări de ordin general. În tabelul 2.2 sînt date și expresiile logice ale acestor semnale fără ca în schema din fig. 2.5 să fie indicate și schemele logice corespunzătoare.

Avînd în vedere că numărul porturilor de intrare-ieșire din interfețele analizate în acest capitol a fost mic, s-a adoptat așa-numita „selecție liniară" adică asociind cîte un bit de adresă pentru un anume port, ceea ce conduce la o schemă logică mai simplă. Atunci cînd această cale nu este satisfăcătoare se procedează la decodificarea rangurilor de adresă prin porți sau cu circuite specializate de tipul 8205, *CDB* 442, *SN* 74154 ș.a. în funcție de necesitățile impuse de aplicația în cauză.

TABELUL 2-2 ADRESELE DE SELECȚIE PENTRU PORȚURILE DE INTRARE ȘI IEȘIRE

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	ADR. HEXA	MNEM		EXPLICAȚII
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001	SCR1	⎍	INCARCA OCTET1IN REG. TAMPON AL MINIIMPRIMANTEI
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002	SCR2	⎍	INCARCA OCTET 2 IN REG. TAMPON AL MINIIMPRIMANTEI
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	8004	SCR3	⎎	VALIDEAZĂ IEȘIRI REG. TAMPON MINIIMPRIMANTA
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	8008	SCR4	⎎	INHIBA IESIRI REG. TAMPON MINIIMPRIMANTA
1	0	0	0	0	1/0	0	0	0	0	0	1	0	0	0	0	81/010	SCR5	⎎	SCRIE ($\overline{WR}$=0) IN 8251, DATE (A10=0) SAU COMENZI (A10=1)
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	8020	SCR6		
																	⋮		NE FOLOSITE
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000	SCR14	⎎	SCRIE LA NIVELUL PROGRAMATORULUI PTR. 2708
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001	CIT 1	⎍	CITIRE STARE R SI T DE LA MINIIMPRIMANTA
1	0	0	0	0	1/0	0	0	0	0	0	1	0	0	0	0	81/010	CIT 2	⎎	CITIRE DIN 8251; DATE CIND A10=0 SAU STARE CIND A10=1
																	⋮		NE FOLOSITE
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000	CIT 14		CITIRE LA NIVELUL PROGRAMATORULUI PTR 2708

$S = A15 \cdot (\overline{Wo}) \cdot (\overline{WR})$

SCR 1 = $S \cdot A_0$
SCR 2 = $S \cdot A_1$
SCR3 = $\overline{S \cdot A_2}$
SCR4 = $\overline{S \cdot A_3}$
SCR5 = $\overline{S \cdot A_4}$

$C = A15 \cdot MEMR \cdot DBIN$

CIT 1 = $C_{\cdot} A_0$
CIT2 = $\overline{C \cdot A_4}$
CIT 14 = $\overline{C \cdot A_{14}}$

2.7. INTERFAȚA CU MINIIMPRIMANTA SEIKO

Miniimprimanta SEIKO MODEL 104, [2 ; 3] este un dispozitiv de tipărire simplu și ieftin, folosit în calculatoarele de birou *CE* 125T, produse de către fabrica de calculatoare din București. În principiu miniimprimanta este compusă dintr-un tambur pe suprafața căruia se disting 13 generatoare. Pe fiecare generator este gravat același caracter de 13 ori. Caracterele, astfel dispuse pe tambur, sînt cifrele de la 0 la 9, punctul zecimal, semnul minus și un semn special. Înseamnă că imprimanta poate fi folosită numai pentru tipărirea numerelor, pe cel mult 13 coloane. În dreptul fiecărui caracter de pe o generatoare se găsește un ciocănel, deci în total 13 ciocănele. Imprimanta mai conține 13 electromagneți, cîte unul pentru acționarea fiecărui ciocănel, un electromagnet pentru avansul hîrtiei și a panglicii tușate și un electromagnet pentru ridicarea panglicii, în cazul tipăririi în roșu. Tamburul imprimantei se rotește cu viteză constantă, antrenat fiind cu un motoraș electric.

Pentru a putea realiza tipărirea unui număr este necesar să se știe care generatoare de pe tambur se găsește, la un moment dat, în fața ciocănelelor. În acest scop imprimanta generează două semnale, notate cu *R* și *T*, prin intermediul unor traductoare inductive (fig. 2.12. a, b). Pe un disc se găsește fixat un magnet permanent care va induce într-o bobină semnalul *R* (fig. 2.12.d) la fiecare rotație completă a tamburului. Magnetul permanent este astfel poziționat pe axul tamburului încît semnalul *R* indus apare atunci cînd în dreptul ciocanelor se găsește generatoarea pe care se găsesc gravate caracterele zero. De la un al doilea traductor inductiv se obține semnalul *T*, care în raport cu *R*,

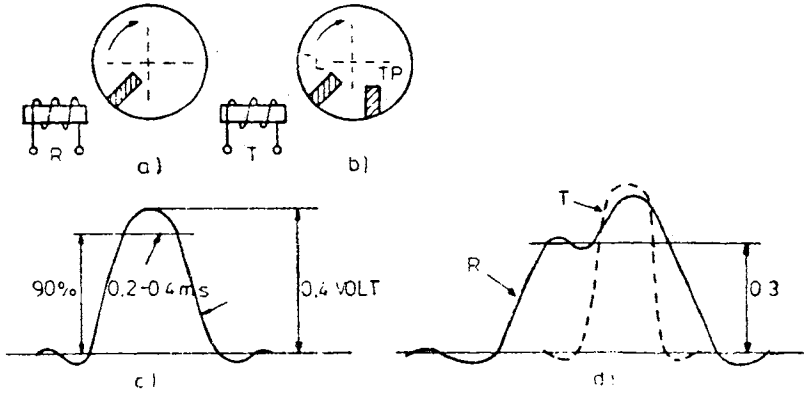


Fig. 2.12. Generarea semnalelor R și T de către mecanismele miniimprimantei

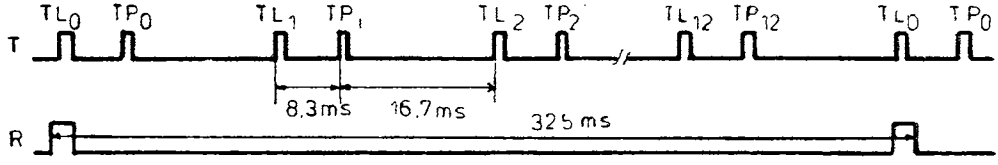


Fig. 2.13. Diagrama de timp a semnalelor R și T

după formarea acestora cu semnale *TTL*, apare așa cum se indică în fig. 2.13. În semnalul *T* se disting perechi de semnale, notate cu *TL* și *TP*, care provin de la cei doi magneți permanenți fixați pe același disc (fig. 2.13. b). semnalele notate cu *TL* apar atunci cînd în fața ciocănelelor se găsește cîte o generatoare cu caractere. Prin numărarea semnalelor *TL* din momentul apariției semnalului *R* rezultă care generatoare se găsește în fața ciocănelelor. Intervalul de timp dintre semnalele TL_i și TP_i este rezervat pentru a decide dacă în numărul ce urmează a fi tipărit, există caractere ce coincid cu cele aflate în fața ciocănelelor. De exemplu, în cazul numărului 828184, cînd în fața ciocănelelor se va găsi generatoarea cu cifra 8, va rezulta coincidența în coloanele 2, 4 și 6, numerotînd coloanele de la dreapta spre stînga. În intervalul de timp dintre TP_i și TL_{i+1} se vor alimenta numai electromagneții ciocanelor din acele coloane unde în procesul de comparare a rezultat o coincidență. Procedînd astfel rezultă că la o rotație completă a tamburului cu caractere numărul va fi în întregime tipărit, deoarece în acest timp toate caracterele existente pe tambur au trecut prin fața ciocanelor.

Pentru tipărirea unui nou rînd este necesar ca mai întîi să se facă un avans al hîrtiei. Avansul sau deplasarea hîrtiei are loc dacă timp de 25 milisecunde, adică pe durata dintre TL_i și TL_{i+1}, se alimentează electromagnetul care provoacă avansul hîrtiei. După alimentarea electromagnetului de avans trebuie să se aștepte cca 75 ms, ca hîrtia să ajungă efectiv în noua poziție și numai după acest timp se poate începe tipărirea unui nou rînd. Dacă se dorește ca un rînd să fie tipărit în roșu, atunci mai întîi se va alimenta timp de 25 milisecunde, electromagnetul ce provoacă ridicarea panglicii tușate pe poziția de tipărire în roșu. Se așteaptă cca 75 ms pentru ca panglica să ajungă efectiv în poziția necesară și după aceea se poate comanda tipărirea rîndului. De reținut că la comanda de avans a hîrtiei panglica tușată revine în mod automat în poziția de tipărire în negru.

2.7.1. SCHEMA INTERFEȚEI CU MINIIMPRIMANTA

Pentru satisfacerea cerințelor descrise mai sus, referitoare la modul cum trebuie să fie comandată imprimanta pentru tipărirea unui rînd, există mai multe variante posibile de concepere a interfeței dintre imprimantă și micro-

calculator. În fig. 2.14 se prezintă o soluție a cărei particularitate constă în aceea că pentru tipărirea unui rînd, microcalculatorul este dedicat în întregime acestui proces, în sensul că acesta va prelucra în cadrul unor subprograme informațiile recepționate de la imprimantă și va lua deciziile necesare pentru acționarea electromagneților ciocănelelor la momentele potrivite.

Urmărirea modului de funcționare a interfeței se va face pentru cazul tipăririi unui singur număr cu maximum 12 cifre zecimale, memorate în 12 adrese succesive din *RAM* în codul *BCD*.

Semnalele generate de către miniimprimantă, R_R și T_R, sînt amplificate și formate ca semnale *TTL* așa cum se arată în fig. 2.14, circuitele respective fiind montate pe o plăcuță așezată în apropierea imprimantei. Starea semnalelor *R* și *T* transmise la schema de interfață, este încărcată în bistabilele B_R și B_T, sincronizate cu semnalul *SYNC* (fig. 2.5).

Microprocesorul 8080 cercetează tot timpul starea semnalelor *R* și *T* prin citirea stării bistabilelor *BR* și *BT*. În momentul cînd constată că $R=T=1$, înseamnă că tamburul începe o nouă rotație și că în fața ciocănelelor se află generatoarea pe care sînt gravate cele 13 caractere zero. În acest moment are loc saltul la subprogramul în care se pregătește informația ce urmează a fi încărcată în registrele 8212(1) și 8212(2) sub forma „*OCTET* 1" și respectiv „*OCTET* 2". Această informație va fi numită pe scurt „mască

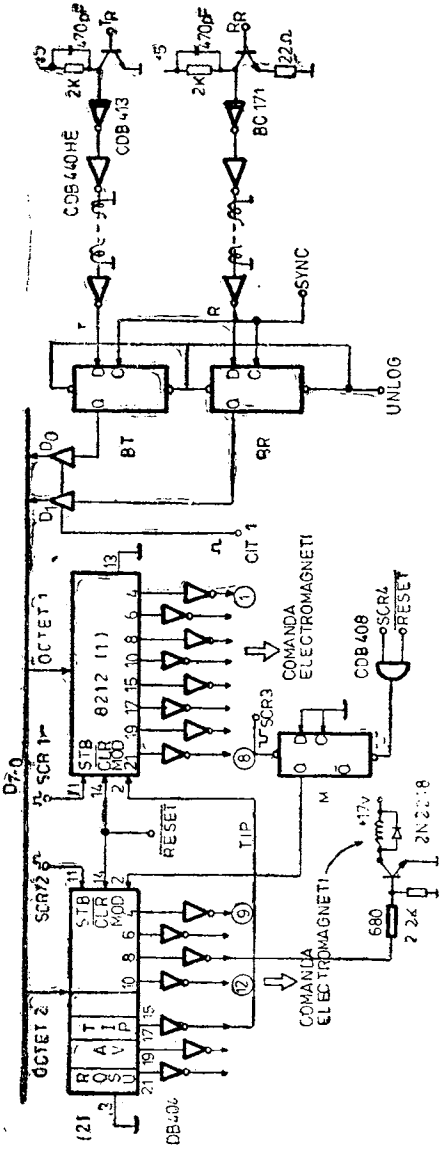


Fig. 2.14. Schema interfeței cu miniimprimanta

pentru tipărire" și se obține în felul următor. Se citește din *RAM* cifra cea mai puțin semnificativă a numărului, aceea care urmează a fi tipărită pe coloana numărul 1 și se compară dacă este zero. În caz afirmativ atunci bitul cel mai puțin semnificativ din *OCTET* 1 devine egal cu zero și în caz contrar bitul respectiv devine egal cu 1. Se citește din *RAM* următoarea cifră zecimală din cadrul numărului, se testează dacă este zero, și se completează în mod corespunzător următorul bit din *OCTET* 1. Se continuă procesul pînă cînd se completează în întregime *OCTETUL* 1, moment în care acesta se încarcă în 8212 (1) cu o instrucție de tipul *STA* 8001, care generează semnalul *SCR* 1.

În continuare se trece la completarea măștii ce revine *OCTETULUI* 2. Primii patru biți din *OCTETUL* 2 se completează în mod similar, ei rezultînd din compararea ultimelor patru cifre zecimale (cele mai semnificative) din cadrul numărului. Al 5-lea bit din *OCTET* 2 corespunde ciocănelului din coloana a 13-a pe care s-a presupus că nu apar cifre tipărite și din acest motiv întotdeauna va avea starea zero. Al 6-lea bit din *OCTET* 2, notat *TIP*, se completează cu 0 atunci cînd urmează o tipărire. Al 7-lea bit din *OCTET* 2, notat *AV*, este folosit pentru comanda electromagnetului de avans a hîrtiei și al 8-lea notat Roșu este folosit pentru comanda electromagnetului de ridicare a panglicii pe roșu.

Pentru tipărire ultimii biți din *OCTET* 2 se vor completa astfel : $ROSU = AV = 1$ și $TIP = 0$. Se încarcă *OCTET* 2 astfel format, din acumulator în 8212(2) cu o instrucție de tipul *STA* 8002, care generează semnalul *SCR* 2. Formarea și încărcarea măștii în cele două registre este obligatoriu să se facă în intervalul dintre TL_0 și TP_0. În continuare microprocesorul va trece la citirea stării semnalelor *R* și *T* pentru a identifica momentul apariției semnalului TP_0. Cînd constată că $R = 0$ și $T = 1$, ceea ce înseamnă $TP_0 = 1$, poziționează bistabilul *M* în starea 1 cu instrucția STA 8004, care generează semnalul *SCR* 3. Pînă în acest moment *M* s-a găsit în starea zero și din acest motiv ($MOD = 0$ la 8212) ieșirile registrului 8212(2) s-au găsit în starea de impedanță ridicată, echivalentă cu unu logic pentru intrările invertoarelor *CDB* 404. Ieșirile invertoarelor comandă alimentarea electromagneților ciocănelelor de tipărire, dar pentru că toate ieșirile se găsesc în starea zero nici un electromagnet nu este alimentat. Semnalul *MOD* pentru registrul 8212(1) rezultă că se găsește de asemenea în starea zero și deci în starea zero se vor găsi și ieșirile invertoarelor legate la 8212(1). Cînd bistabilul *M* trece în starea 1 atunci semnalul *MOD* pentru 8212(2) devine 1 și la ieșirile invertoarelor se transmite sub formă negată starea biților din *OCTET* 2. Cum bitul *TIP* a fost programat a fi egal cu zero înseamnă că semnalul *MOD* pentru 8212(1) devine egal cu 1 și la ieșirile invertoarelor se transmite starea negată a biților din *OCTET* 1. Dacă bitul în mască a fost 0 atunci electromagnetul din coloana respectivă va fi alimentat. Întrucît $ROSU = AV = 1$ înseamnă că electromagneții respectivi nu vor fi alimentați. Microprocesorul 8080 după poziționarea pe 1 a bistabilului *M* va cerceta din nou starea semnalelor *R* și *T* pentru a identifica apariția semnalului

TL_1. Cînd constată că $R=0$, $T=1$, adică $TL=1$, atunci poziționează bistabilul M în starea zero executînd instrucția STA 8008, care generează semnalul SCR 4. Ieșirile registrelor 8212 trec în starea de impedanță ridicată și în consecință se va întrerupe alimentarea electromagneților. Acum în dreptul ciocănelelor se găsește generatoarea pe care sînt gravate cifre de 1 și întreg procesul descris mai sus, de formare a unei noi măști și de alimentare a electromagneților, se repetă. Se procedează la fel pînă la generatoarea a 12-a inclusiv. A 13-a generatoare conține un semn special (săgeată orizontală) care nu poate să apară în cadrul unui număr zecimal. După tipărirea în întregime a rîndului urmează avansul hîrtiei pentru a pregăti tipărirea rîndului următor. În acest scop se formează *OCTET* 2 în care numai bitul *AV* este egal cu 0 restul fiind în starea 1, inclusiv bitul *TIP*. Se încarcă *OCTET* 2 în 8212(2) fără a fi nevoie să se actualizeze starea existentă în 8212(1). Se comandă poziționarea bistabilului M în starea unu ceea ce are ca efect alimentarea numai a electromagnetului pentru avansul hîrtiei. Microprocesorul 8080 intră într-o subrutină de întîrziere pe care o părăsește după 25 msec și atunci repune bistabilul M în starea zero, întrerupînd alimentarea electromagnetului de avans. În mod similar se procedează dacă se dorește ridicarea panglicii pentru tipărirea în roșu.

Descrierea făcută mai sus a urmărit în primul rînd să clarifice rolul și modul de comportare a circuitelor din cadrul interfeței sub acțiunea comenzilor generate de către microcalculator. Se face acum observația că cele două registre tip 8212 pot fi înlocuite cu 4 registre tip *CDB* 495. În acest caz ieșirile registrelor 495 vor constitui intrări la porți de tip *SI*, validitate de către bistabilul M. Cînd M se găsește în starea zero ieșirile porților vor fi în starea zero și cînd $M=1$ la ieșirile porților se va transmite starea registrelor 495. În fig. 2.14 s-a indicat numai pentru ciocănelul din coloana 11 modul de alimentare a electromagnetului.

2.7.2. PROGRAMELE NECESARE TIPĂRIRII

Este indicat ca programele pentru tipărire să fie concepute astfel încît să satisfacă cerințele impuse de o anumită aplicație. Programele valabile pentru orice gen de aplicație ocupă un spațiu mult prea mare de memorie pentru sistemele în configurație redusă, ca cel luat în considerare aici. Pe de altă parte este bine ca programele să fie organizate modular încît să asigure o flexibilitate cît mai mare în stabilirea formatului de tipărire și să se poată izola ușor zonele care urmează a fi modificate cînd se schimbă aplicația.

În continuare se vor da unele indicații privind un mod posibil de organizare a programelor pentru o aplicație în care a fost necesar să se tipărească un rînd

în roșu, să se avanseze hîrtia peste un rînd, să se tipărească 8 rînduri de date și apoi să se lase trei rînduri libere. Setul de programe se compune din :

a) Subprogramul *TIPRIND*, lung de 133 octeți care realizează următoarele :

— urmărirea stării imprimantei, pentru a identifica începutul unei rotații a tamburului și apoi apariția unei noi generatoare cu caractere în fața ciocănelelor ;

— pregătirea măștii de tipărire pentru fiecare generatoare cu caractere ;

— comanda electromagneților ciocănelelor ;

— avansul hîrtiei la rîndul următor cînd s-a terminat tipărirea unui rînd.

b) subprogramul *ABRIR* lung de 52 octeți care realizează următoarele :

— comandă alimentarea electromagnetului pentru ridicarea panglicii pe roșu ;

— comandă alimentarea electromagnetului pentru avansul hîrtiei cu un rînd.

Selectarea uneia din cele două funcții se face printr-un parametru transmis de programul apelant în registrul A al microprocesorului.

c) subprogramul *DELAY* 25, lung de 12 octeți, care realizează prin numărare o întîrziere de 25 milisec, necesară pentru alimentarea, pe durata prescrisă, a electromagneților :

d) programul principal, lung de 62 octeți, pregătește într-o zonă din *RAM* informația ce urmează a fi tipărită pe un rînd și apelează subprogramul *TIPRIND*. Toate lungimile de programe indicate mai sus se referă la partea de cod ; nu s-au luat în considerare zonele de date folosite de fiecare program.

În Tabelul 2.3 se prezintă o porțiune din subprogramul *TIPRIND*. Registrele microprocesorului sînt folosite astfel :

— registrul *B* indică care generatoare (cifră) se află în fața ciocănelelor;

— registrul *C* indică numărul de cifre dintr-un rînd ce au fost deja comparate cu cifra de pe generatoare ;

— registrul *E* servește la formarea măștii ;

— registrele *H*, *L* indică adresa curentă din zona tampon a imprimantei rezervată în *RAM*.

Inițial registrul *E* este pus pe zero. Apoi se aduc succesiv din zona tampon în registrul A cifrele care trebuie tipărite pe un rînd. Se compară cifra adusă cu conținutul registrului *B*, deci cu cifra din fața ciocănelelor. Dacă cifrele nu coincid se adună 1 la registrul *E*. Aceasta va avea ca efect punerea pe unu a celei mai puțin semnificative cifre din *E*, întrucît înaintea adunării, ultima cifră din *E* este pusă pe zero. După actualizarea conținutului registrelor *C* și *H*, *L* se verifică dacă au fost deja comparate 4 caractere. În caz afirmativ acestea vor fi cele tipărite spre marginea din dreapta a rîndului, așa că ele se trimit în *OCTET* 2 adăugîndu-li-se în față și biții pentru comanda tipăririi (*ROSU*=*AVANS*=1 *TIP*=0). Registrul *E* se repune pe zero, fiind astfel pregătit pentru formarea măștii pentru ultimele 8 coloane ale rîndului.

După fiecare comparare, cu excepția celor pentru coloanele 4 și 12, are loc rotirea cu o poziție spre stînga a conținutului registrului *E*. În program, înainte de a executa instrucția *RAL* se pune fanionul de depășire *C* pe zero, astfel că la capătul din dreapta al registrului *A*, și apoi *E*, se va introduce în urma rotirii cifra zero. După compararea ultimei cifre din zona tampon, se încarcă masca *OCTET* 1 în 8212(1). În continuare subprogramul *TIPRIND* conține instrucțiile necesare pentru comanda tipăririi. Se citește starea semnalelor *R* și *T* urmărind în ce stare se găsește *T*. Avînd în vedere că *T* este un impuls relativ lat, după formarea măștii se cercetează mai întîi dacă TL_i a ajuns în starea zero și în caz afirmativ se așteaptă în continuare ca TP_i să devină 1, moment în care se alimentează electromagneții ciocănelelor. Alimentarea electromagneților se menține pînă la apariția lui TL_{i+1} moment în care începe formarea următoarei măști de tipărire. Procedînd astfel nu este indicat ca microprocesorul să părăsească execuția subprogramelor de tipărire datorită unei cereri de întrerupere, deoarece în acest caz există posibilitatea de a pierde un semnal *T*, sau altfel spus sincronizarea cu tamburul și tipărirea se va face greșit.

Un asemenea mod de lucru poate fi acceptat numai atunci cînd în aplicația dată rezultă că pe durata tipăririi microprocesorul nu este solicitat să facă față și altor sarcini și ar rămîne oricum într-o stare de așteptare. În caz contrar deservirea imprimantei de către microprocesorul 8080 trebuie să se facă prin intermediul întreruperilor, ceea ce implică o altă organizare a setului de subprograme precum și unele modificări în schema de interfață. Sugerăm aici o cale ce poate fi urmată în această situație.

Semnalele *R* și *T* să conducă fiecare la generarea unei cereri de întrerupere pe nivele diferite de tratare. În subrutina de tratare a întreruperilor de tip *T* evidența semnalelor *TL* și *TP* din semnalul *T* se validează numai după apariția unei întreruperi de tip *R*. Este posibil de asemenea ca semnalul *T* să fie separat în componentele *TL* și *TP* fiecare generînd o cerere de întrerupere proprie.

Avînd în vedere că pot exista și alte cereri de întrerupere și că operația de tipărire ar putea să fie necesară numai la intervale relativ mari de timp, este indicat ca microprocesorul să valideze influența semnalelor *R* și *T* numai în momentul cînd poate începe operația de tipărire. Dacă operațiile de tipărire apar la intervale mari de timp, atunci pentru a evita uzura mecanismelor imprimantei, aceasta va fi alimentată sub comanda microprocesorului numai la momentul potrivit. Se menționează că tamburul ajunge la turația nominală din momentul alimentării în cca 0,2 s. După ce tipărirea s-a terminat se va comanda întreruperea alimentării imprimantei.

Tabelul 2.3.

Subprogramul TIPRIND

Adresă	Cod hexa	Mnemonic	Comentarii
08 00	AF	XRA A	0→A
08 01	4F	MOV C,A	0→C
08 02	5F	MOV E,A	0→E
08 03 08 04 08 05	21 00 02	LXI H, TAMPON	Se consideră că adresa de început a tamponului imprimantei în RAM este 0200
08 06	7E	MOV A,M	Citește următorul caracter
08 07	B8	CMP B	Compară cu cifre de pe generatoare
08 08 08 09 08 0A	CA 0C 08	JZ NUDEPL	La coincidență se omite incrementarea lui E
08 0B	1C	INR E	(E)+1→E
08 0C	0C	INR C	(C)+1→C
08 0D	23	INX H	(H,L)+1→H, L
08 0E	79	MOV A,C	
08 0F 08 10	FE 04	CPI	S-au comparat 4 caractere ?
08 11 08 12 08 13	C2 1F 08	JNZ TEST 12	Dacă nu, salt la testul (C)=12
08 14 08 15 08 16	7B F6 E0	MOV A,E ORI	Adaugă biții de comandă
08 17 08 18 08 19	32 02 80	STA 8002	Memorează OCTET 2 în 8212 (2)
08 1A 08 1B	1E 00	MVI	0→E
08 1C 08 1D 08 1E	C3 06 08	JMP MUTA	Continuă compararea

Tabelul 2.3. (Continuare)

Adresă	Cod hexa	Mnemonic	Comentarii
08 1F 08 20	FE 0C	CPI	(C)=12 ?
08 21 08 22 08 23	CA 2C 08	JZ TIPAR	Dacă da, s-a terminat compararea
08 24	7B	MOV A,E	Rotește E cu o poziție la stînga. Rotirea se poate face numai în A. Se pune CARRY=0
08 25	37	STC	
08 26	3F	CMC	
08 27	17	RAL	
08 28	5F	MOV E,A	
08 29 08 2A 08 2B	C3 06 08	JMP MUTA	Continuă compararea
08 2C	7B	MOV A,E	Memorează OCTET 1 în 8212 (1)
08 2D	32	STA 8001	
08 2E 08 2F	01 80		
08 30 08 31 08 32	3A 01 80	LDA 8001	Citește stare R și T pînă cînd rezultă că T=0. Se verifică astfel dacă TL_i a ajuns în starea zero. Este posibil ca formarea măștii să se termine în timpul cît TP_i este încă în starea unu
08 33 08 34	E6 01	ANI	
08 35 08 36	F6 01	CPI	
08 37 08 38 08 39	CA 30 08	JZ ASTZERO	
08 3A 08 3B 08 3C	3A 01 80	LDA 8001	Se citește din nou R și T pînă cînd rezultă că T=1, ceea ce înseamnă că TP_i=1. În acest moment urmează a se alimenta electromagneții
08 3D 08 3E	E6 01	ANI	
08 3F 08 40	FE 01	CPI	
08 41 08 42 08 43	C2 3A 08	JNZ ASTUNU	

Tabelul 2.3. (Continuare)

Adresă	Cod hexa	Mnemonic	Comentarii
08 44 08 45 08 46	32 04 80	STA 8004	Se comandă alimentarea electromagneţilor prin poziţionarea bistabilului M pe unu
08 47 08 48 08 49	3A 01 80	LDA 8001	Se citeşte starea lui R şi T pînă cînd rezultă că T=0 ceea ce se înseamnă TP_t=0
08 4A 08 4B	E6 01	ANI	
08 4C 08 4D	FE 01	CPI	
08 4E 08 4F 08 50	CA 47 08	JZ STALIM	
08 51 08 52 08 53	3A 01 80	LDA 8001	Se citeşte starea lui R şi T pînă cînd rezultă că T=1, ceea ce înseamnă că TL =1 moment în care urmează a se întrerupe alimentarea electromagneţilor
08 54 08 55	E6 01	ANI	
08 56 08 57	FE 01	CPI	
08 58 08 59 08 5A	C2 51 08	JNZ $STALIM_1$	
08 5B 08 5C 08 5D	32 08 80	STA	Se comandă întreruperea alimentării electromagneţilor prin poziţionarea pe zero a bistabilului M
08 5E	04	INR B	Programul se reia de la început pînă cînd se ajunge la a 12 generatoare
08 5F	78	MOV A,B	
08 60 08 61	FE 0C	CPI	
08 62 08 63 08 64	C2 00 08	JNZ DOI	
08 65 08 66	3E DF	MVI A, DF	Se încarcă în 8212(2) octetul DF, unde AV=1
08 67 08 68 08 69	32 02 80	STA 8002	

Tabelul 2.3 (continuare)

Adresă	Cod hexa	Mnemonic	Comentarii
08 6A 08 6B 08 6C	32 04 80	STA 8004	Se comandă alimentarea electromagnetului de avans a hîrtiei
08 6D 08 6E 08 6F	CD — —	CALL DELAY25	Se apelează subrutina de întîrziere de 25 m sec.
08 70 08 71 08 72	32 08 80	STA 8008	Se comandă întreruperea alimentării electromagnetului de avans.

2.8. INTERFAȚA CU DISPOZITIVUL DE AFIȘARE DAF 1001

Circuitul USART 8251, din familia circuitelor anexă la microprocesorul 8080, este un circuit cu posibilități multiple, destinat transmiterii și recepționării de date în format serie și în regim sincron sau asincron. Cu acest circuit se poate realiza o legătură în mod simplu cu dispozitivul de afișare *DAF* 1001. Regimurile de funcționare ale circuitului 8251 se presupun a fi cunoscute.

Modul de conectare al circuitului 8251 la microcalculatorul prezentat în fig. 2.5 este indicat în fig. 2.15. Circuitul 8251 este tratat de către 8080 ca un periferic și selectat prin linia de adresă $A4$, cînd aceasta are valoarea unu logic. Altfel spus, 8251 în operațiile de citire sau scriere este privit ca o locație de memorie cu adresa 8010 ($A15=A4=1$). Pentru a distinge, la nivelul lui 8251, datele de cuvintele de comandă sau de informația de stare, se folosește ca și semnal C/D linia de adresă $A10$. Cînd $A10=0$, operația de scriere sau citire se referă la date și adresarea lui 8251 se face cu adresa 8010. Cînd $A10=1$ operația de scriere sau citire se referă la cuvinte de comandă, respectiv la informația de stare și adresarea lui 8251 se face cu adresa 8110. Microcalculatorul în operațiile de citire generează impulsul *CIT* 2 și în operațiile de scriere impulsul *SCR* 5, indicate și în tabelul 2.2.

Transferul datelor cu ajutorul lui 8251 se poate realiza prin intermediul întreruperilor sau prin program. În primul caz dacă 8251 a terminat de recepționat un caracter sau dacă caracterul anterior a fost complet transmis, se poziționează pe unu logic semnalele de ieșire $T \times RDY$, respectiv $R \times RDY$, care pot fi folosite pentru a genera cereri de întrerupere la 8080. În al doilea caz, 8080 cercetează prin program starea lui 8251 și în funcție de aceasta decide dacă poate trimite un nou caracter spre 8251, respectiv dacă poate prelua caracterul recepționat de către 8251 de pe linie.

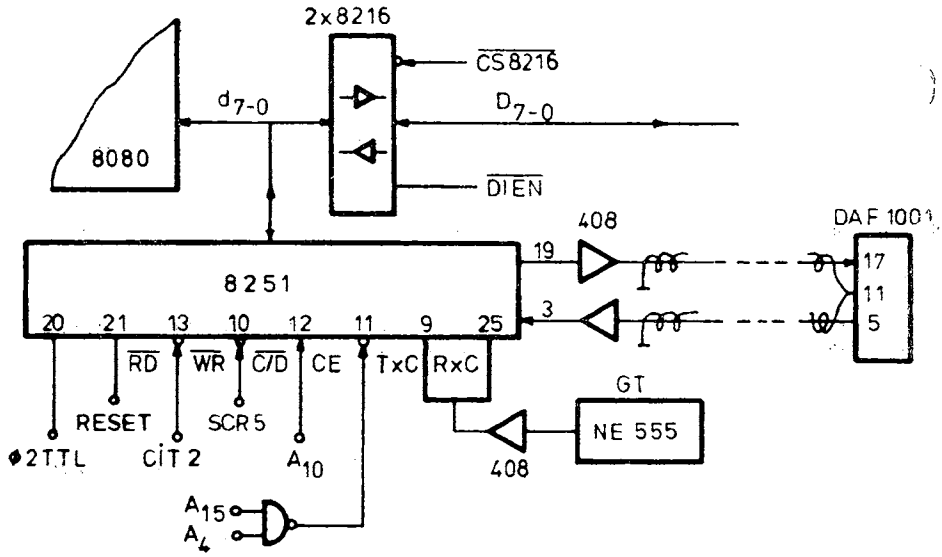


Fig. 2.15. Conectarea circuitului USART 8251 la microcalculator

În schema din fig. 2.15 semnalele $T \times RDY$ și $R \times RDY$ nu sînt folosite întrucît s-a adoptat soluția ca transferul de date să se facă prin program și nu prin intermediul întreruperilor.

Referitor la conectarea circuitului 8251 la magistrala de date a microcalculatorului se face următoarea observație. Magistrala de date a circuitului 8251 poate fi încărcată de asemenea numai cu o singură sarcină *TTL*. Din acest motiv magistrala de date a circuitului 8251 este legată direct la magistrala de date a microprocesorului. Expresiile semnalelor $\overline{CS_{8216}}$ și $\overline{DIEN}$ date în cap. 2.4 trebuie să se modifice în forma :

$$\overline{CS_{8216}} = HLDA + MEMR.DBIN.A_4.A_{15};$$
$$\overline{DIEN} = DBIN.$$

În acest fel, cînd 8080 realizează o operație de citire la nivelul lui 8251, porțile de ieșire spre 8080 din circuitul 8216 trec în starea de impedanță ridicată, fiind validate numai porțile de ieșire din 8251.

2.8.1. PROGRAMAREA CIRCUITULUI USART 8251.

Înainte de a executa orice fel de operație asupra lui 8251, inclusiv cea de citire a stării, este necesar ca 8251 să fie programat. Aceasta înseamnă că după comanda de inițializare, transmisă prin semnalul *RESET*, stă se încarce în

8251 cuvîntul *MOD* și după aceea cuvîntul *COMANDA*. În cuvîntul *MOD* se specifică regimul în care se dorește să lucreze 8251 : sincron sau asincron, lungimea caracterului, numărul de biți în semnalul *STOP* etc. De reținut că „lungimea caracterului" nu se referă și la bitul de paritate și că acesta va fi întotdeauna adăugat de către 8251 în regimul de transmitere. Prin cuvîntul de *COMANDA* se asigură ca 8251 să funcționeze conform cu regimul indicat în cuvîntul *MOD*. În acest caz este de reținut faptul că indicatorul $R \times E$, din cuvîntul de *COMANDA*, are numai rolul de a valida sau inhiba posibilitatea poziționării pe unu logic a semnalului $R \times RDY$, care apare la terminalul 14 al capsulei și că nu se referă la operația propriu-zisă de recepție a caracterelor, operație ce se desfășoară indiferent de starea lui $R \times E$. Datorită acestui fapt, și anume că partea de receptor din 8251 funcționează tot timpul, este posibil ca 8251 să conțină date incorecte în momentul validării semnalului $R \times RDY$. Din acest motiv se recomandă ca 8251 să fie „descărcat" de caracterul pe care îl conține, imediat după ce în cuvîntul de comandă s-a prevăzut $R \times E = 1$. Această operație de citire de date se poate face fără a aștepta ca indicatorul $R \times RDY$ din informația de stare să devină unu logic.

De remarcat, de asemenea, că trimiterea unui cuvînt de *COMANDA* spre 8251 în timp ce acesta nu a terminat încă cu trimiterea unui caracter în serie pe linie, poate conduce la perturbarea procesului de conversie din paralel în serie. Pentru a evita acest lucru, un nou cuvînt de comandă se va încărca în 8251 numai după ce se constată că $T \times RDY$ s-a poziționat în starea unu logic.

Subprogramele de dialog cu *DAF* 1001 depind de modul de transmisie, în regim caracter sau mesaj și de formatul în care se dorește a se realiza afișarea. Din acest motiv se vor prezenta două programe simple pentru transmisie și recepție urmînd ca utilizatorul să-și dezvolte programele necesare conform cerințelor impuse de aplicația concretă.

Prin programul dat în tab. 2.4 se realizează trimiterea repetată către *DAF* 1001 a caracterului *A*. S-a considerat că programul respectiv va fi executat imediat după comanda *RESET* și începe cu încărcarea cuvintelor *MOD* și *COMANDA* în 8251. Deoarece în cuvîntul *MOD* s-a prevăzut $B_2 = 1$ și $B_1 = 0$, frecvența generatorului de tact (*GT*) realizat cu astabilul *NE* 555, trebuie să fie de 16 ori mai mare decît rata de transfer exprimată în biți pe secundă. De exemplu, pentru o rată de transfer de 110 biți/sec, frecvența generatorului de tact va fi de 1760 Hz. Din octetul de date primit de la microcalculator, 8251 va reține șapte biți (D_{6-0}) pentru codul caracterului și va completa pe al 8-lea (D_7) ca bit de paritate pară. Pe linie, 8251 va trimite caracterul în formatul 8/11, adică precedat de semnalul *START*, cu durata egală cu a unui bit și urmat de semnalul *STOP*, cu durata egală pentru doi biți.

Un astfel de program este util în faza de punere la punct pentru că în regim automat semnalele la nivelul lui 8251 variind periodic pot fi urmărite cu osciloscopul. De exemplu, vizualizînd cu osciloscopul semnalul $T \times D$ (terminalul 19 de la 8251) pe ecran trebuie să apară unda binară specifică carac-

Tabelul 2.4.

Program pentru transmitere

Adresa	Cod Hexa	Mnemonic	Explicații
00 01	3E FA	MVI A	MOD=FA→Acumulator S2=S1=EP=PEN=L2=B2=1 ; L1=B1=0
02 03 04	32 10 84	STA	MOD→8251 $A_{15}=A_{10}=A_4=1$; se generează SCR5
05 06	3E 11	MVI A	COMANDA=11→Acumulator EH=IR=RTS=SBRK=RyE=DTR=0 IR=T×EN=1
07 08 09	32 10 84	STA	COMANDA→8251 $A_{15}=A_{10}=A_4=1$; se generează SCR5
0A 0B	3E 41	MVI A	Cod caracter A→Acumulator
0C 0D 0E	32 10 80	STA	Date→8251 Cod caracter A→8251 $A_{15}=A_4=1$; $A_{10}=0$; se generează SCR5
0F 10 11	3A 10 84	LDA	STARE 8251→Acumulator $A_{15}=A_{10}=A_4=1$; se generează CIT2
12 13	E6 01	ANI	T×RDY în acumulator
14 15 16	E2 0A 00	JPO	Dacă T×RDY=1 se reia cu trimiterea unui nou caracter A
17 18 19	C3 0F 00	JMP	Dacă T×RDY=0 se reia cu citirea stării

terului *A*. Cu ajutorul panoului programul se poate lansa în execuție în regimurile instrucție cu instrucție sau ciclu cu ciclu.

Un program simplu, pentru verificarea funcționării corecte numai în regim de recepție, este dat de tabelul 2.5. Programul este considerat independent de cel analizat anterior și se consideră memorat în *RAM* începînd de la adresa hexazecimală 0000. În tab. 2.5 în coloana *ADRESA* s-a trecut numai octetul mai puțin semnificativ al adresei. Programul fiind lansat în execuție

Tabelul 2.5.

Program pentru recepție

Adresa	Cod hexa	Mnemonic	Explicații
00 01	3E FA	MVI A	MOD=FA→Acumulator $S_2=S_1=EP=PEN=L_2=B2=1$
02 03 04	32 10 84	STA	MOD→8251 $A_{15}=A_{10}=A_4=1$; se generează SCR5
05 06	3E 14	MVI A	COMANDA=14→Acumulator ER=R×E=1
07 08 09	32 10 84	STA	COMANDA→8251 $A_{15}=A_{10}=A_4=1$; se generează SCR 5
0A 0B 0C	21 00 01	L×I H	ADRESA RAM→H, L
0D 0E 0F	3A 10 80	LDA	DATE 8251→Acumulator Se descarcă 8251 $A_{10}=C/\overline{D}=0$
10 12 13	3A 10 84	LDA	STARE 8251→Acumulator $A_{10}=C/\overline{D}=1$; se generează CIT2
14 15	E6 02	ANI	În acumulator rămîne R×RDY
16 17 18	EA 10 00	JPE	Dacă R×RDY=0 salt la adresa 0010, ptr. citirea stării
19 1A 1B	3A 10 80	LDA	DATE 8251→Acumulator $A_{15}=A_4=1$; $A_{10}=C/\overline{D}=0$ Se generează CIT2
1C	77	MOV M,A	(A)→Memorie
1D	23	INX H	+1→(H,L)
1E 1F 20	3C 10 00	JMP	Salt la citirea stării pentru recepția următorului caracter

automată, dacă se constată că în *RAM* nu se memorează caracterele acționate la tastatura *DAF*-ului, atunci prin intermediul panoului se poate trece la execuția programului în regimul ciclu cu ciclu (vezi cap. 2.5.2). În acest regim se urmărește dacă totul decurge normal pînă cînd se ajunge la instrucția *LDA* 8010, memorată la adresa 0019. Pentru această instrucție, prin acționarea butonului *STEP*, se parcurg ciclurile *M*1, *M*2, *M*3 și ne oprim în ciclul *M*4. În această situație dacă se acționează o tastă de la *DAF*, pe *LED*-urile ce afișează liniile de date trebuie să apară codul caracterului acționat.

2.9. INTERFAȚA CU MONITORUL TV TEHNOTON

Pentru microcalculatorul descris se va prezenta în continuare o modalitate de afișare pe ecranul unui monitor TV, fabricat de către întreprinderea Tehnoton. Folosirea monitoarelor TV sau a receptoarelor TV normale, ca echipamente de afișare a datelor este larg răspîndită [4].

2.9.1. PROBLEME GENERALE PRIVIND MODUL DE FUNCȚIONARE

Un caracter poate fi afișat pe un ecran TV cu ajutorul unei matrici de 5×7 puncte așa cum se arată în fig. 2.16. În timpul baleiajului pe orizontală spotul este modulat în intensitate astfel încît punctele luminoase pe ecran să apară acolo unde în fig. 2.16 sînt marcate punctele negre. Un rînd de caractere cuprinde 10 linii TV, dintre care prima linie este nefolosită, următoarele 7 linii servesc la conturarea caracterului prin puncte și ultimele două linii TV sînt de asemenea nefolosite și servesc la formarea distanței între rînduri.

Între două caractere din același rînd se lasă o distanță egală cu cea necesară pentru trei puncte Pe o linie TV sînt deci necesare $5+3=8$ puncte pentru un caracter și pentru 32 de caractere pe un rînd, rezultă că sînt necesare $32 \times 8 = 256$ puncte. Perioada oscilatorului de baleiaj pe orizontală, *TH*, este de 64 microsecunde, din care cursa directă (vizibilă) durează cca 52 microsecunde, restul timpului fiind consumat pentru readucerea spotului la începutul următoarei linii. Dacă în 52 microsecunde, cît durează partea vizibilă a unei linii, spotul trebuie modulat în intensitate de 256 de ori, rezultă că frecvența de modulare a spotului trebuie să fie aproximativ 5 MHz ($52/256 = 0{,}2\mu s$). La sfîrșitul liniei 320, adică după $320 \times 64 = 20$ milisecunde, generatorul de baleiaj pe verticală comandă revenirea spotului la începutul primei linii. În acest fel cadrele se repetă de 50 de ori într-o secundă, evitîndu-se fenomenul de pîlpîire pentru ochiul uman.

Se analizează în continuare care ar trebui să fie principiul de lucru al interfeței cu microcalculatorul avînd în vedere că informația de vizualizat se

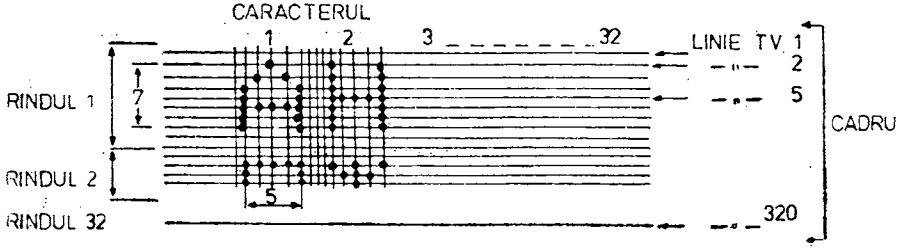


Fig. 2.16. Baleiajul TV și reprezentarea caracterelor prin puncte

găsește în memoria *RAM* a sistemului. Din fig. 2.16 în primul rînd se desprinde concluzia că fiecare caracter trebuie memorat nu printr-un singur cod binar, ci sub forma imaginii sale de pe ecran. Pentru caracterele indicate în fig. 2.16 ar fi necesar ca la adresa x să se memoreze octetul 00001000, ce corespunde primei linii din caracterul *A*, la adresa $x+1$ să se memoreze octetul 00100010, ce corespunde primei linii din caracterul *H*, și în mod analog pentru următoarele 30 de caractere din primul rînd. La adresa $x+32$ se va memora octetul 00010100 ce corespunde celei de a doua linii din caracterul *A*, la adresa $x+33$ se va memora octetul 00100010 ce corespunde celei de a doua linii din caractetul *H* și la adresele următoare octeții corespunzători celei de a doua linii pentru următoarele caractere din primul rînd. În timpul afișării, cînd spotul descrie linia a 2-a TV, octetul de la adresa x se va încărca într-un registru și cifra binară ce rezultă prin deplasări succesive la dreapta va servi la modularea în intensitate a spotului. Dacă cifra binară este zero, spotul va fi blocat și dacă cifra binară este unu spotul va provoca iluminarea ecranului. Deplasarea octetului în registru se va face cu un semnal de tact cu frecvența de 5 MHz. După șapte semnale de tact ultima cifră din stînga ajunge pe prima poziție din dreapta. La al 8-lea semnal de tact, deci după $8\times0{,}2=1{,}6$ microsecunde, în registru se va încărca octetul memorat la adresa $x+1$; acesta prin deplasări la dreapta va modula în intensitate spotul și pe ecran vor apare cele două puncte luminoase din prima linie a caracterului *H*. Din cele de mai sus rezultă că încărcarea registrului de deplasare, cu un nou octet citit din memoria *RAM*, trebuie să se facă tot la intervale de 1,6 μs. O asemenea viteză de citire a memoriei *RAM* nu poate fi asigurată de către microprocesorul 8080 întrucît numai execuția instrucției *OUT*, de exemplu, durează 5 microsecunde. Din această cauză se impune ca citirea octeților din memorie să se facă de către schema de interfață prin procedeul de acces direct la memorie (*DMA*). În acest scop interfața va genera semnalul *HOLDM*, care va aduce microprocesorul în starea *HOLD* stare în care 8080 cedează controlul asupra magistralelor de adresă și de date. În această stare interfața preia controlul memoriei *RAM* și pe toată durata vizualizării 8080 rămîne inactiv.

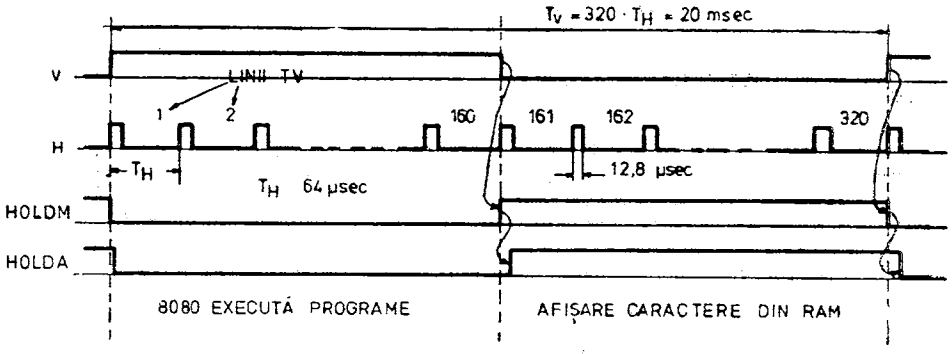


Fig. 2.17. Folosirea memoriei RAM de către 8080

Numărul caracterelor afișate pe ecranul monitorului TV este de $32 \times 32 = 1\,024$ caractere. Dacă se reduce numărul caracterelor afișate la jumătate, adică la 16 rînduri a 32 caractere fiecare, atunci se creează posibilitatea ca pe durata unei jumătăți de cadru (10 milisecunde) microprocesorul să lucreze și în cealaltă jumătate să rămînă în starea *HOLD*, pentru a permite afișarea celor 16 rînduri de caractere. În fig. 2.17 se arată modul în care trebuie să varieze în acest caz semnalul *HOLDM* generat de către interfață. Pe durata primelor 160 de linii 8080 deține controlul asupra memoriei *RAM* și pe durata următoarelor 160 de linii 8080 se află în starea *HOLD*, controlul asupra memoriei *RAM* fiind deținut de către interfața de afișare cu monitorul TV.

Caracterele în memoria *RAM* se memorează, de regulă, sub forma unui singur octet (codul *ASCII* sau *EBCDIC*) și nu sub forma imaginii lor prin puncte pe ecran, așa cum s-a considerat pînă acum. Este posibil totuși ca informația ce urmează a fi vizualizată să fie recodificată prin program în această formă. Acest lucru se realizează mai simplu cu ajutorul unui circuit integrat de tip memorie *ROM* numit „generator de caractere". În fig. 2.18 se prezintă modul de lucru al generatorului de caractere tip *TMS* 2501. Fiecare caracter este memorat sub forma unei matrici 5×7. Intrările D_{5-0} servesc pntru adresarea unui caracter (matrice) din cele 64 memorate și intrările r_{3-1} pentru selecția unei linii din matrice. Astfel, dacă la intrările D_{5-0} se poziționează adresa 01 (în octal) și intrările $r_3r_2r_1$ se găsesc în starea zero, atunci ieșirile B_{5-1} se găsesc toate în starea zero. Pentru aceeași adresă de intrare, dacă $r_3r_2r_1$ trec în starea 001, atunci numai ieșirea B_3 va fi în starea unu. Din fig. 2.18 rezultă că prin modificarea intrărilor r_{3-1}, la ieșirile B_{5-1} se obțin cele șapte linii ale matricei caracterului *A*.

Cu aceste precizări făcute, legăturile principale ale schemei de interfață cu microcalculatorul se prezintă ca în fig. 2.19. Pe durata cît $V = HOLDM = 1$ interfața poziționează adresele pe magistrala de adrese și primește pe magis-

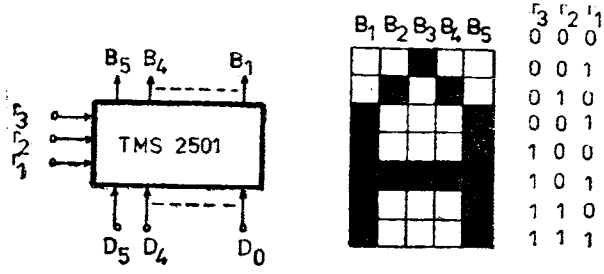


Fig. 2.18. Generatorul de caractere

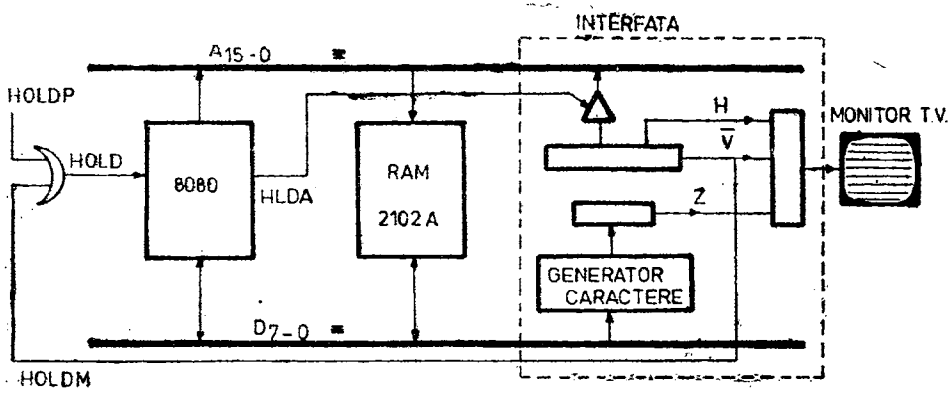


Fig. 2.19. Legăturile principale ale microcalculatorului cu interfața TV

trala de date codul caracterelor din memoria *RAM*, care pe durata afișării trebuie să fie tot timpul selectată și în regim de citire. Ieșirile generatorului de caractere se încarcă în registrul cu deplasare obținîndu-se astfel semnalul *Z*. Din semnalele *Z*, *H* și *V*, generate de către circuitele interfeței, se va obține semnalul video complex pentru monitorul TV.

2.9.2. SCHEMA INTERFEȚEI CU MONITORUL TV

Schema bloc a interfeței cu monitorul TV este prezentată în fig. 2.20. Numărătorul orizontal, format din numărătoarele *N*01 și *N*02, prin divizarea frecvenței semnalului de tact, va genera un semnal *H* cu frecvența $f_H = 15\,625$ Hz, care va servi la sincronizarea oscilatorului de baleiaj pe orizontală din cadrul monitorului. Numărătorul vertical, format din numărătoarele *NV*1 și *NV*2 va genera semnalul *V*, cu frecvența $f_v = 50$ Hz, care va servi la sincro-

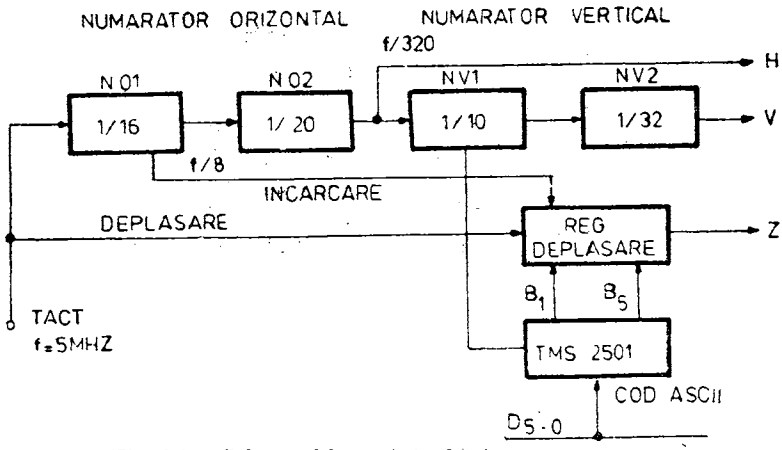


Fig. 2.20. Schema bloc a interfeței cu monitorul TV

nizarea oscilatorului de baleiaj pe verticală din cadrul monitorului. Din numărătorul *N*01 se derivă semnalul *ÎNCĂRCARE*, cu ajutorul căruia starea ieșirilor generatorului de caractere se încarcă în registrul de deplasare tot la al 8-lea semnal de tact. Cele șapte semnale de tact, care apar între două semnale *ÎNCĂRCARE*, comandă deplasarea spre dreapta a octetului existent în registru.

Din numărătorul *NV*1 se derivă semnalele r_{3-1} pentru selecția liniilor din matricea adresată din generatorul de caractere.

Schema în detaliu a interfeței este prezentată în fig. 2.21. Urmărirea modului de funcționare se poate face pe baza explicațiilor date anterior. Unele completări se impun totuși a fi făcute mai ales pentru cîteva aspecte mai de detaliu, dar importante pentru o funcționare corectă. Un rol esențial îi revine semnalului numit *ÎNCĂRCARE* a cărui formare se va analiza urmărind și diagramele din fig. 2.22. Dacă comutatorul bipozițional notat cu X se găsește în poziția indicată în fig. 2.21 atunci, într-o primă aproximație, intrarea *J* a bistabilului *MOD* (*J*/*MOD*) se găsește în starea unu atunci cînd $\overline{V}=1$ și $\overline{H}=1$. Acest lucru înseamnă că poziționarea pe unu a bistabilului *MOD* este posibilă numai în timpul cît spotul descrie liniile TV cu numărul 161 pînă la 320 ($\overline{V}=1$) și numai în timpul cursei directe a spotului, adică în partea vizibilă a liniei TV, cînd $\overline{H}=1$. Sincronizarea bistabilului *MOD* se realizează cu semnalul *C*1. Din fig. 2.22 rezultă că la începutul fiecărei linii TV semnalul *ÎNCĂRCARE* desenat punctat nu se poate genera, deoarece pe durata cursei inverse intrarea *J*/*MOD* s-a aflat în starea zero. Din acest motiv într-o linie TV se vor genera numai 31 de impulsuri *ÎNCĂRCARE*. La fiecare impuls *ÎNCĂRCARE* în registrele *CDB* 495 se va înscrie starea ieșirilor B_{5-1} ale generatorului de caractere. La prima linie TV intrările de selecție r_{3-1} se vor găsi în

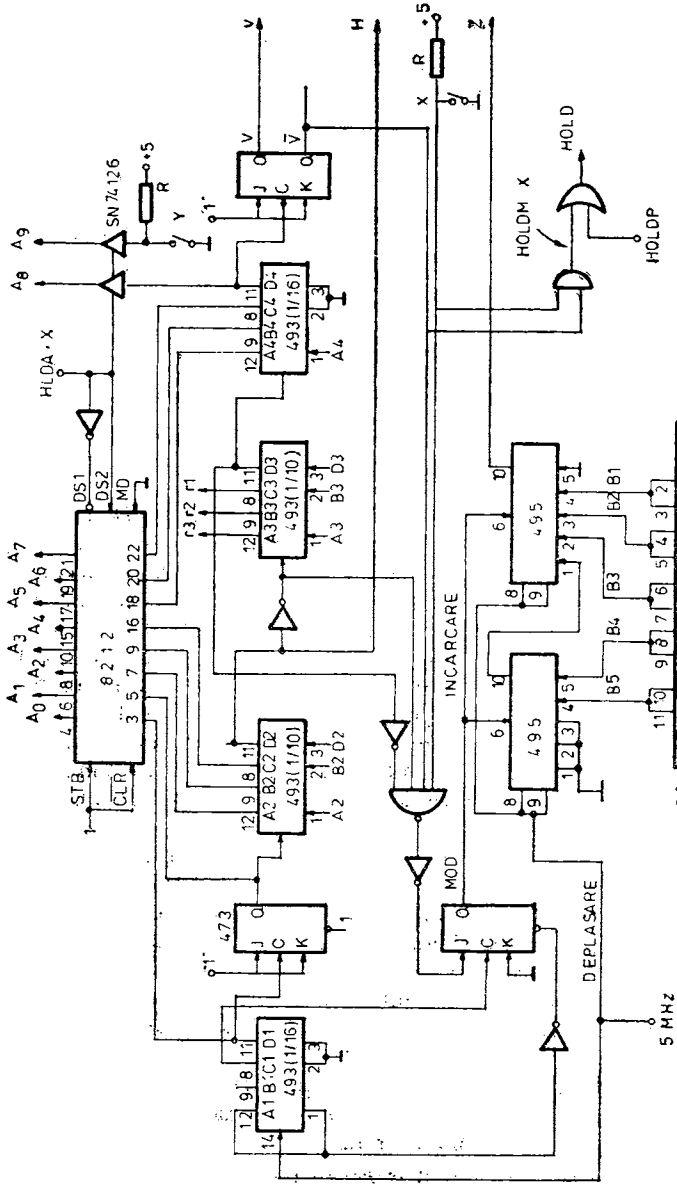


Fig. 2.21. Schema interfeței cu monitorul TV

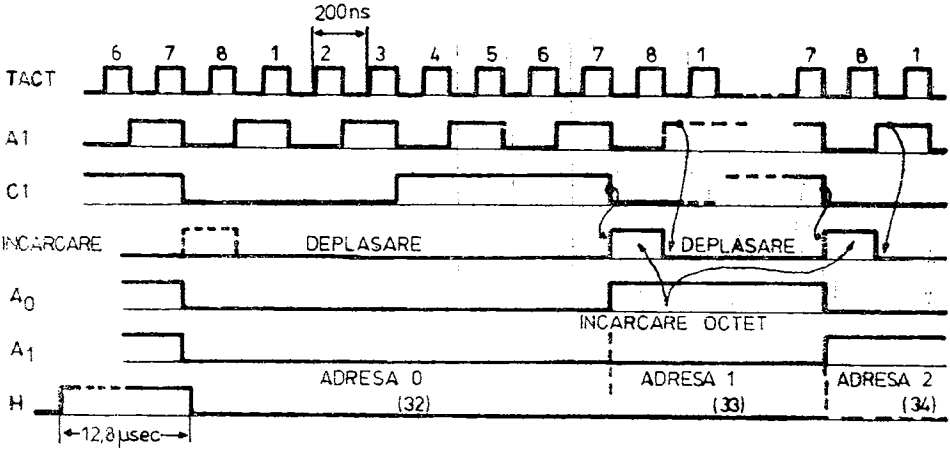


Fig. 2.22. Formarea semnalului „INCARCARE"

starea zero și după cum rezultă din fig. 2.18 în acest caz ieșirile B_1 la B_5 se găsesc toate în starea zero, indiferent de adresa prezentă la intrările D_{5-0}. Din acest motiv pe prima linie din zona vizualizată, care de fapt are numărul 161 nu va apare nici un punct luminos. Pe toată durata următoarei linii (linia 162) intrările de selecție $r_3\, r_2\, r_1$ se vor găsi în starea 001 și pentru toate caracterele primului rînd generatorul de caractere va genera la ieșirile sale linia a 2-a din matrice de 5×7 puncte. În timpul liniilor 169 și 170 ieșirea D_3 a numărătorului *CDB* 493, conectat ca divizor cu 10, se găsește în starea unu și din această cauză în timpul cît sînt descrise aceste linii nu se pot genera impulsurile de încărcare. Linia 171, fiind prima linie din cadrul rîndului 2, nu va conține de asemenea puncte luminoase și prin urmare se formează astfel o distanță între rînduri egală cu trei linii TV.

O altă chestiune importantă ce trebuie discutată este cea referitoare la timpul de acces al memoriei *RAM* și al generatorului de caractere. Din fig. 2.22 rezultă că suma dintre cei doi timpi de acces trebuie să fie mai mică decît 1,6 microsecunde. Timpul de acces al generatorului de caractere *TMS* 2501 este de aproximativ 0,3 microsecunde și al memoriei *RAM*, realizată cu capsule 2102 A, este de 0,35 microsecunde. Pentru memorii *RAM* cu timp de acces mai lung schema trebuie astfel modificată încît să se afișeze un număr corespunzător mai mic de caractere într-un rînd.

Cîteva observații se impun a fi făcute și asupra semnalelor *HOLD* și *HLDA*. Din fig. 2.21 rezultă că semnalul *HOLD* spre microprocesorul 8080 se poate genera fie de la panoul comenzilor manuale cînd *HOLDP*=1 fie de către interfața TV cînd comutatorul *X* se găsește în poziția indicată în figură. Ca răspuns la semnalul *HOLD* microprocesorul activează semnalul *HLDAX*.

În fig. 2.5 se vede că *HLDAX* este resincronizat în exteriorul microprocesorului pentru a obține semnalul *HLDA*, cel folosit și în schema din fig. 2.21. Resincronizarea lui *HLDAX* se impune din următorul motiv. Microprocesorul 8080 poziționează în starea unu logic semnalul *HLDAX*, ca răspuns la *HOLD*, în momente diferite, după cum execută un ciclu de citire sau un ciclu de scriere. Prin datele de catalog ale lui 8080 rezultă că *HLDAX* este poziționat în starea unu cu o întîrziere maximă $t_{DC}=120$ ns, față de frontul pozitiv al semnalului Ø1. În timpul unui ciclu de citire, dacă în *T2* bistabilul interior *HOLD F/F* al microprocesorului este poziționat în starea unu, atunci Ø1 din starea ce urmează lui *T3* poziționează pe unu semnalul *HLDAX* cu întîrzierea t_{DC}. În timpul unui ciclu de scriere, dacă în starea *T2* bistabilul intern *HOLD F/F* al microprocesorului este poziționat în starea unu, atunci Ø1 din starea *T4* poziționează pe unu semnalul *HLDAX*. Magistrala de adresă și cea de date intră în starea de impedanță ridicată cu o întîrziere maximă $t_{FD}=120$ nsec. față de frontul pozitiv al semnalului $\varnothing 2$. Din acest motiv dacă *HLDAX* ar fi folosit în chip de *HLDA* atunci rangurile de adresă A_0 la A_9, generate de către interfață TV, se vor poziționa pe magistrala de adrese a microprocesorului înainte ca aceasta să fi trecut în starea flotantă. Mai ales la sistemele care lucrează cu o frecvență de tact mai mică de 2 MHz ($tcY>500$ ns) acest lucru poate conduce, în timpul unui ciclu de citire sau de scriere, la prelucrarea de către 8080 a unui octet de la o adresă ce diferă de cea pozițională de către 8080 pe magistrala de adrese. Prin sincronizarea lui *HLDAC* cu Ø2 *TTL* se elimină aceste posibile situații de funcționare incorectă.

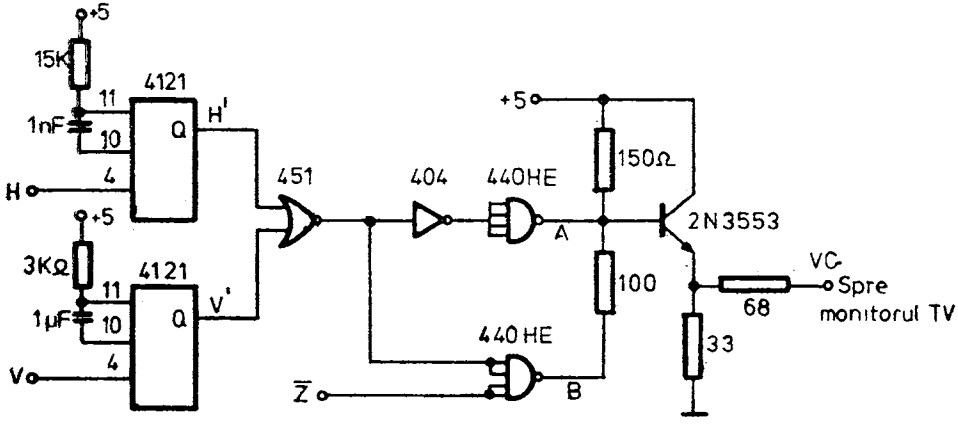


Fig. 2.23. Formatorul semnalului video complex

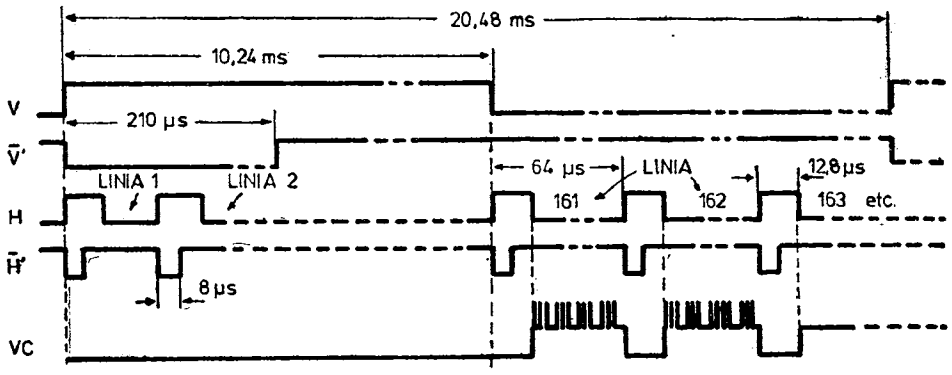


Fig. 2.24. Forma semnalului video complex

Semnalul video complex se obține cu schema din fig. 2.23, a cărei funcționare să fie urmărită și cu ajutorul diagramelor din fig. 2.24.

Expresiile logice ale semnalelor notate cu A și B sînt următoarele :

$$A=\overline{H'}.\overline{V'}\,; \quad B=\overline{\overline{H'}.\overline{V'}.\overline{Z}}$$

Se observă că din moment ce $\overline{V'}=1$ există posibilitatea ca spotul să fie modulat prin intermediul lui Z, dar atîta timp cît $V=1$ semnalul Z este egal cu zero deoarece nu se pot genera semnalele *ÎNCĂRCARE*. Duratele în timp ale semnalelor H' și V' indicate în fig. 2.24 sînt cele măsurate pentru valorile componentelor din fig. 2.23.

Cu modificări neesențiale schema de interfață din fig. 2.21 poate fi folosită și pentru afișarea unui număr de 64 caractere într-un rînd. Aceste modificări se referă la modul de formare al semnalelor de *ÎNCĂRCARE* și *DEPLASARE* conform schemei din fig. 2.25. De asemenea adresele A_0 și A_9 generate de către schema de interfață trebuie deplasate cu un rang la stînga. Rangul $A5$ din adresă se va obține de la ieșirea $D2$, $A6$ de la ieșirea $A4$ a numărătorului *CDB* 493, respectiv rangul A_9 se va obține de la ieșirea D_4 a aceluiași numărător. În fotografia 2.3 se prezintă o imagine pe ecranul TV în regimul de afișare cu 31 caractere pe rînd. Textul respectiv a fost introdus în memoria *RAM*, manual de la panou, după execuția în prealabil a unui program prin care în zona *RAM* afișată s-a înregistrat caracterul blanc.

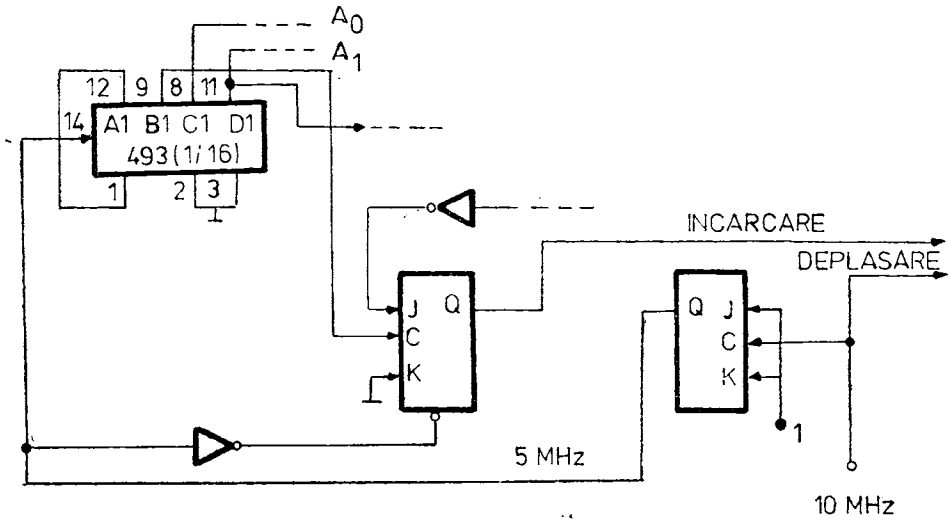


Fig. 2.25. Formarea semnalelor „INCARCARE" şi „DEPLASARE" pentru afişarea a 64 caractere pe un rînd

Foto 2.3. Un text din memoria RAM afişat pe ecranul monitorului TV

2.10. PROGRAMATOR PENTRU MEMORIA EPROM 2708 CONDUS DE MICROCALCULATOR

Datele de catalog [1] fac precizările necesare privind amplitudinea şi modul de variaţie în timp a semnalelor ce intervin la programarea memoriei 2708. De reţinut, conform indicaţiilor de catalog, este faptul că în regim de reprogramare adresele trebuie parcurse succesiv, începînd de la adresa 0 pînă la adresa 1023 de N ori, unde N se calculează în relaţia $N \cdot TPW \geqslant 100$. Cu TPW s-a

notat durata impulsului de programare, care poate fi cuprinsă între 0,1 pînă la 1 milisec. De exemplu, pentru $TPW=0{,}5$ m s rezultă $N=200$, ceea ce înseamnă că de cel puțin 200 de ori memoria trebuie parcursă de la adresa 0 la 1023, prezentînd pentru fiecare adresă, la intrările de date, octetul ce urmează a fi înregistrat la adresa respectivă.

2.10.1. SCHEMA BLOC A PROGRAMATORULUI

Pentru programarea memoriei 2708 este necesar un echipament special, denumit „programator" care are rolul să asigure toate condițiile impuse prin foaia de catalog în regimul de programare. Un astfel de programator poate fi conceput fie ca un echipament de sine stătător, fie ca o extindere a posibilităților oferite de un microcalculator. În ultima variantă circuitele necesare a fi adăugate pentru a realiza operația de programare sînt mai reduse.

În fig. 2.26 se prezintă schema bloc a programatorului pentru memoria 2708, în situația în care pentru microcalculator programatorul apare ca și cum ar fi un echipament de intrare-ieșire selectat atunci cînd $A_{15}=A_{14}=1$.

Blocul numit „Convertor nivele și baza de timp" are două regimuri de funcționare, selectate manual printr-un comutator bipozițional. În poziția „programare" acesta sub comanda microcalculatorului va genera nivelele de tensiune și duratele în timp necesare pentru semnalele ce intervin în regimul de programare. Microcalculatorul are sarcina de a genera adresele succesive din memoria 2708 și de a poziționa pe liniile de date D_{7-0}, pentru fiecare adresă, octetul de date ce urmează a fi înregistrat la adresa respectivă. După parcurgerea în întregime a memoriei 2708 procesul se repetă de N ori. Aceste operații sînt asigurate prin execuția de către microcalculator a unui subprogram, denu-

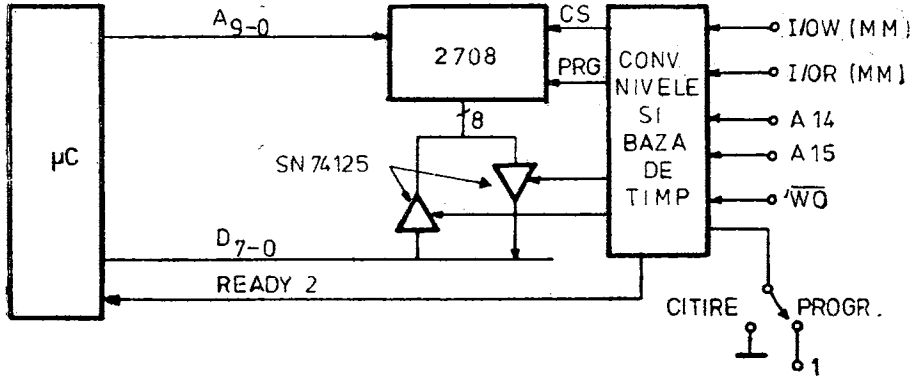


Fig. 2.26. Schema bloc a programatorului

mit „subprogram pentru programarea memoriei 2708". Acest subprogram trebuie să fie înregistrat într-o memorie *ROM* a microcalculatorului și apelat atunci cînd se execută operații de programare. Informația ce urmează a fi înregistrată în memoria 2708, supusă regimului de programare, trebuie să se găsească în memoria *RAM* a microcalculatorului.

În lipsa altor echipamente de programare pentru memoria 2708, înseamnă că prima încercare de programare se va referi la înregistrarea în memoria 2708 chiar a „subprogramului de programare". Această informație se va încărca în memoria *RAM*. Tot în memoria *RAM*, în acest caz particular, se va încărca și „subprogramul pentru programarea memoriei 2708".

După ce operația de programare s-a terminat comutatorul bipozițional se trece în poziția *CITIRE*, stare în care microcalculatorul poate realiza operații de citire din memoria deja programată. Un subprogram anume poate fi conceput pentru a verifica corectitudinea programării. Se va discuta ulterior situația în care rezultă că programarea nu s-a executat corect. Se presupune acum că această operație s-a desfășurat corect, caz în care capsula de memorie 2708 poate fi scoasă din programator și introdusă ca un bloc de memorie *ROM* în microcalculator.

2.10.2. CONVERTORUL DE NIVELE ȘI BAZA DE TIMP

Schema de detaliu este dată în fig. 2.27. Dacă comutatorul bipozițional este trecut manual în poziția „programare", atunci semnalul de selecție (*CS*) al memoriei 2708 ajunge la +12 volt așa cum se impune în regimul de programare.

Semnalul $\overline{WO}$, din informația de stare, cînd se găsește în starea zero logic indică faptul că instrucția în curs de execuție va realiza o operație de scriere prin activarea semnalului $\overline{WR}$. Această instrucție va trebui să poziționeze pe liniile de adresă, A_{9-0}, valoarea zero și pe liniile de date D_{7-0} octetul ce urmează a se înregistra în memoria 2708 la adresa zero.

Se presupune acum că registrele *D* și *E* ale microprocesorului au fost inițializate la valoarea *C*000. Dacă în această situație microcalculatorul execută instrucția *STAXD*, atunci în ciclul de mașină *M*2 microprocesorul poziționează pe liniile de adresă A_{15-0} adresa *C*000 ($A_{15}=A_{14}=1$) și pe liniile de date D_{7-0} octetul existent în acumulator. Informația de stare din ciclul *M*2 va conține $\overline{WO}=0$, ceea ce arată că semnalul $\overline{WR}$ va fi activat ($\overline{WR}=0$) la momentul potrivit în vederea realizării operației de scriere. Trecerea din starea unu în starea zero a semnalului $\overline{WO}$ provoacă declanșarea monostabilelor *QMON*1 și *QMON*2 (fig. 2.27). Monostabilul *QMON*1 face ca semnalul *READY*2 să treacă din starea unu în starea zero și din acest motiv 8080 din starea *T*2 a ciclului *M*2 va trece în starea de așteptare *TW*, cu $\overline{WR}$ activ ($\overline{WR}=0$), stare în care rămîne pînă cînd *QMON*1 revine în starea inițială.

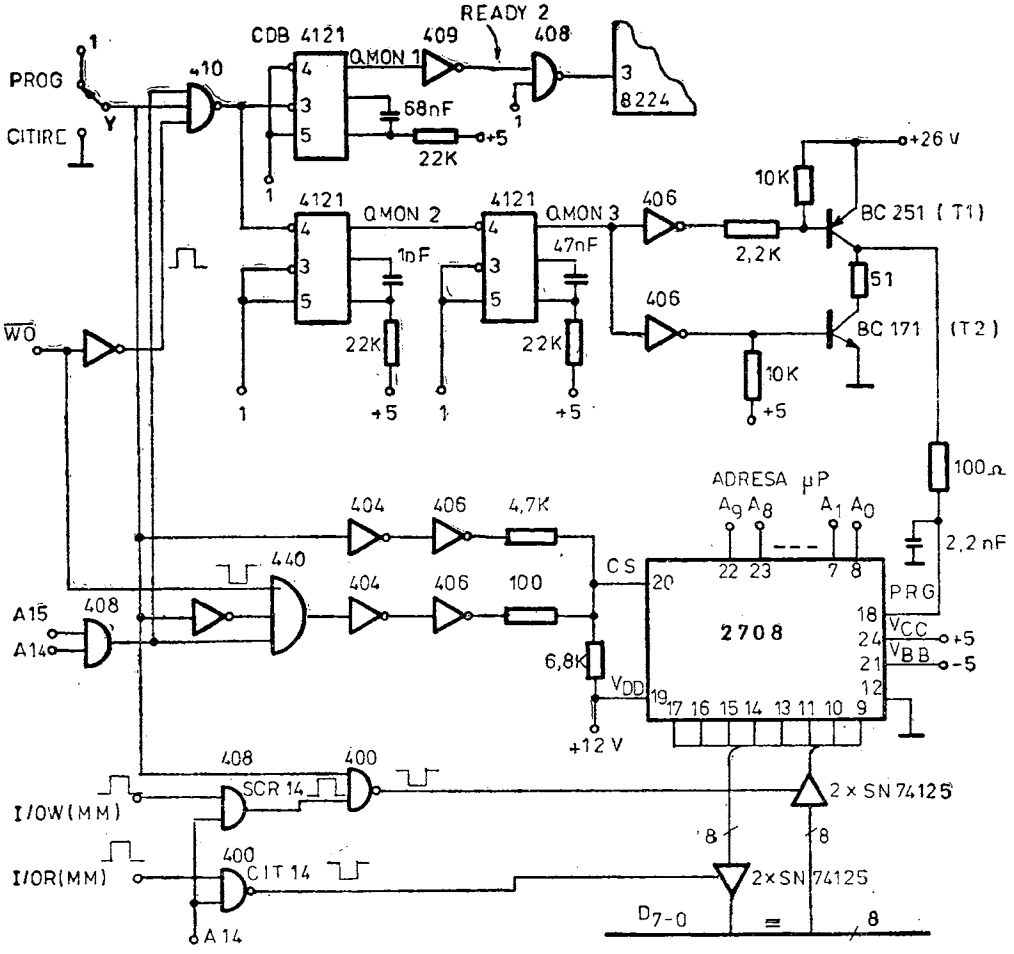


Fig. 2.27. Convertorul de nivele şi baza de timp

Monostabilul *QMON*2 are rolul de a introduce o întîrziere de cel puţin de 10 milisecunde, din momentul în care adresa şi datele sînt stabile, pînă la generarea impulsului de programare. Cînd *QMON*2 revine în starea zero se declanşează monostabilul *QMON*3 care în starea unu determină durata impulsului de programare (*TPW*). Cînd *QMON*3=1 tranzistorul *T*2 este blocat şi impulsul de programare, *PRG*, trece de la nivelul 0 la +26 volt şi rămîne la această valoare pînă cînd *QMON*3 revine în starea zero.

Cînd *QMON*1 revine în starea zero, întrucît *READY*2 devine 1, microprocesorul 8080 părăsește starea de așteptare și trece la execuția următoarei instrucții. Procesul se va repeta pentru următoarele adrese după cum rezultă din analiza subprogramului pentru programarea memoriei 2708.

2.10.3. SUBPROGRAM PENTRU PROGRAMAREA MEMORIEI 2708

Subprogramul pentru programarea memoriei 2708 este prezentat cu unele explicații în Tabelul 2.6, considerînd că este înregistrat într-o memorie *ROM* 2708 care ocupă adresele 0800 la *OBFF*. Pentru înțelegerea lui mai ușoară se fac cîteva precizări legate și de lansarea lui în execuție. S-a presupus că memoria *RAM* ocupă adresele 0000 la 03*FF*. Informația ce urmează a fi înregistrată în ROM se introduce în RAM începînd de la adresa 0005 pînă la adresa notată cu H_M, L_M, unde H_M poate ajunge pînă la valoarea 03 și L_M pînă la valoarea *FF*. Operatorul va înregistra în RAM la adresa 0003 valoarea L_M mărită cu o unitate și la adresa 0004 valoarea lui H_M. În *RAM* începînd de la adresa 0000 va introduce instrucția de salt necondiționat la subprogramul de programare (*JMP* 0800). Aceste etape operatorul le parcurge în timp ce microcalculatorul se găsește în starea *HOLD*. În continuare va poziționa comutatorul din fig. 2.27 în poziția *PROGRAMARE*, va acționa butonul *RESET* și va scoate microcalculatorul din starea *HOLD*, eliberînd după aceea butonul *RESET*. În subprogram s-a prevăzut o buclă de întîrziere de cca 1 milisecundă după instrucțiile de tip *STAXD* cu scopul ca o nouă declanșare a bistabilului *QMON*1 să asigure un factor de umplere în jur de 50% pentru semnalul *READY* 2.

Informația ce urmează a fi înregistrată în ROM este preluată de la adrese succesive din RAM, începînd cu adresa 0005 și trimisă spre ROM. Ultimul octet citit din RAM va fi cel de la adresa H_ML_M. În continuare, ca și octeți de date, se vor transfera spre *ROM* octeți de forma *FF*, pînă se ajunge la ultima adresă din ROM (03*FF*). Procesul se va repeta de 256 de ori, numărarea repetărilor făcîndu-se în registrul *C*.

Procesul de programare, avînd în vedere duratele semnalelor din fig. 2.28, durează cca 9 minute. Se trece comutatorul din fig. 2.27 pe poziția *CITIRE*. Etapa următoare ce trebuie parcursă se referă la verificarea corectitudinii programării. Verificarea se poate face cu un subprogram anume sau manual. La verificarea manuală operatorul va întrerupe alimentarea sistemului și va introduce capsula de memorie *ROM* astfel programată, ca un bloc de memorie al microcalculatorului. De reținut că prin întreruperea alimentării, informația din RAM se va pierde. Sistemul se realimentează și prin comenzi de examinare se vor vizualiza adrese succesive din *ROM*. Dacă s-a programat un număr relativ mare de octeți operația este incomodă deoarece durează mult. Din

Tabelul 2.6.

Subprogramul pentru programarea memoriei eprom 2708

Adresa	Cod	Mnemonic	Explicații
0800 0801	0E FF	MVI C	În registrul C se încarcă FF
0802 0803	06 50	MVI B	În registrul B se încarcă 50 în hexa.
0804 0805 0806	11 00 C0	LXI D	În registrul D se încarcă C0 și în E se încarcă 00
0807 0808 0809	21 05 00	LXI H	În registrul H se încarcă 00 și în L se încarcă 05
080A	7E	MOV A,M	Octetul de la adresa existentă în H,L se introduce în A
080B	12	STAX D	Octetul din A se înscrie la adresa existentă în perechea de registre D, E
080C	05	DCR B	(B)−1→B
08 0D	78	MOV A, B	(B)→A
08 0E 08 0F	FE 00	CPI	(A)=00 ?
08 10 08 11 08 12	C2 0C 08	JNZ	Dacă (A)≠0 salt la instrucția de la adresa 08 0C
08 13 08 14	06 50	MVI B	Reinițializarea registrului B
08 15	13	INX D	Adresa din D,E crește cu o unitate
08 16	23	INX H	Adresa din H,L crește cu o unitate
08 17 08 18 08 19	3A 04 00	LDA	Încarcă în A valoarea maximă pentru octetul de adresă din H
08 1A	BC	CMP H	(A)=(H) ?
08 1B 08 1C 08 1D	C2 0A 08	JNZ	Dacă (H)<(A) salt la instrucția de la adresa 080 A

Tabelul 2—6 (continuare)

Adresa	Cod	Mnemonic	Explicații
08 1E 08 1F 08 20	3A 03 00	LDA	Încarcă în A valoarea maximă pentru octetul de adresă din L
08 21	BD	CMP L	(A)=(L) ?
08 22 08 23 08 24	C2 0A 08	JNZ	Dacă (L)<(A) salt la instrucția de la adresa 080 A
08 25 08 26	3E FF	MUI A	Se încarcă FF în A
08 27	12	STAX D	IN ROM se înregistrează FF
08 28	05	DCR B	(B)−1→B
08 29	78	MOV A,B	(B)→A
08 2A 08 2B	FE 00	CPI	(A)=00 ?
08 2C 08 2D 08 2E	C2 28 08	JNZ	Dacă (A)≠00 salt la instrucția de la adresa 0828
08 2F 08 30	06 50	MVI B	Se reinițializează registrul B
08 31	7A	MOV A,D	(D)→A
08 32 08 33	FE C3	CPI	(A)=C3 ?
08 34 08 35 08 36	CA 3B 08	JZ	Dacă octetul superior de adresă ROM a ajuns la limita C3 atunci salt la instrucția de la adresa 083A
08 37	13	INX D	Conținutul perechii de registre D,E crește cu o unitate
08 38 08 39 08 3A	C3 25 08	JMP	Salt necondiționat la instrucția de la adresa 0825
08 3B	7B	MOV A,E	(E)→A
08 3C 08 3D	FE FF	CPI	(A)=FF ?

Tabelul 2— 6 (continuare)

Adresa	Cod	Mnemonic	Explicații
08 3E 08 3F 08 40	C2 37 08	JNZ	Dacă conținutul registrului E nu a ajuns încă la valoarea FF atunci se mărește cu o unitate adresa din perechea de registre, D, E
08 41	0C	INR C	(C)+1→C
08 42	79	MOV A,C	(C)→A
08 43 08 44	FE FF	CPI	(A)=FF ?
08 45 08 46 08 47	C2 04 08	JNZ	Dacă memoria ROM nu s-a parcurs de N ori (aici N=256) procesul se reia
08 48	76	HALT	Oprire

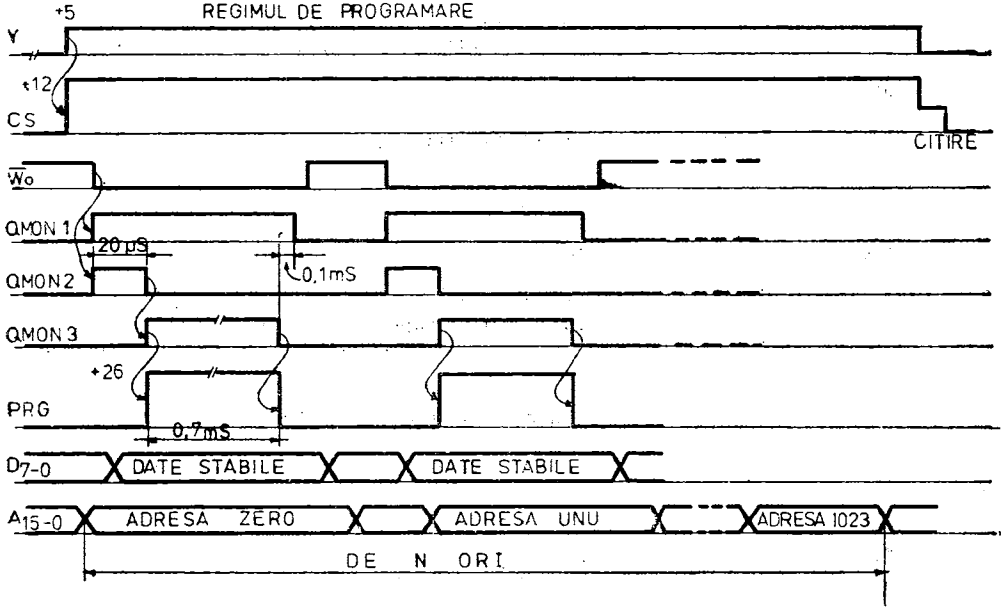


Fig. 2.28. Diagrama semnalelor în regimul de programare

acest punct de vedere verificarea programării cu un subprogram este de preferat. Dacă se constată încă de la început că informația înregistrată în ROM nu este cea corectă atunci este indicat să se repete procesul de programare după ce capsula a fost ștearsă, ceea ce înseamnă aducerea tuturor octeților în strea *FF* [1].

Este indicat ca înainte de a trece la o nouă programare să se verifice dacă într-adevăr după ștergere fiecare octet din ROM este de forma *FF*. Se vor nota, dacă este cazul, adresele la care octeții au o altă configurație. În cazul că numărul acestora nu este prea mare, capsula încă mai poate fi folosită, făcînd uz de zonele corecte. Dacă și la următoarele încercări de programare rezultatul nu este corect, atunci capsula este defectă.

Pe măsură ce noi programe sînt puse la punct, se va proceda la înregistrarea lor în memorii ROM 2708. Astfel presupunem că într-o capsulă ROM 2708 s-a înregistrat un program în zona de adrese 0 la 650_{10}. După un timp un nou program a fost pus la punct și se dorește ca și acesta să fie înregistrat începînd de la adresa 651. În acest caz se poate proceda în felul următor. Se introduce în programator capsula de memorie ROM 2708 programată și cu un subprogram se copiază din aceasta în RAM, începînd de la adresa 0005, informația existentă în zona de adrese 0 la 650. În continuare, de la panou sau de la tastatura hexazecimală, se introduce în *RAM* noul program. După actualizarea valorilor pentru H_M și L_M se poate lansa în execuție subprogramul de programare.

2.11. PROGRAMATOR AUTONOM PENTRU MEMORIA EPROM 2708

Există numeroase situații în care este avantajos ca să se dispună de un programator de memorii EPROM care să funcționeze autonom fără a apela pentru comanda lui la un sistem cu microprocesor. O variantă posibilă pentru un astfel de programator, realizabilă cu mijloace relativ simple este prezentată în continuare.

Schema bloc din figura 2.29 redă principalele elemente ale programatorului precum și modul de interconectare al acestora. Programatorul poate efectua următoarele operații :

a. introducerea, prin tastatură sau bandă perforată, a informației în memoria RAM ;

b. verificarea, adresă cu adresă, a conținutului memoriei RAM ;

c. programarea automată a memoriei EPROM ;

d. verificarea, adresă cu adresă, a conținutului memoriei EPROM ;

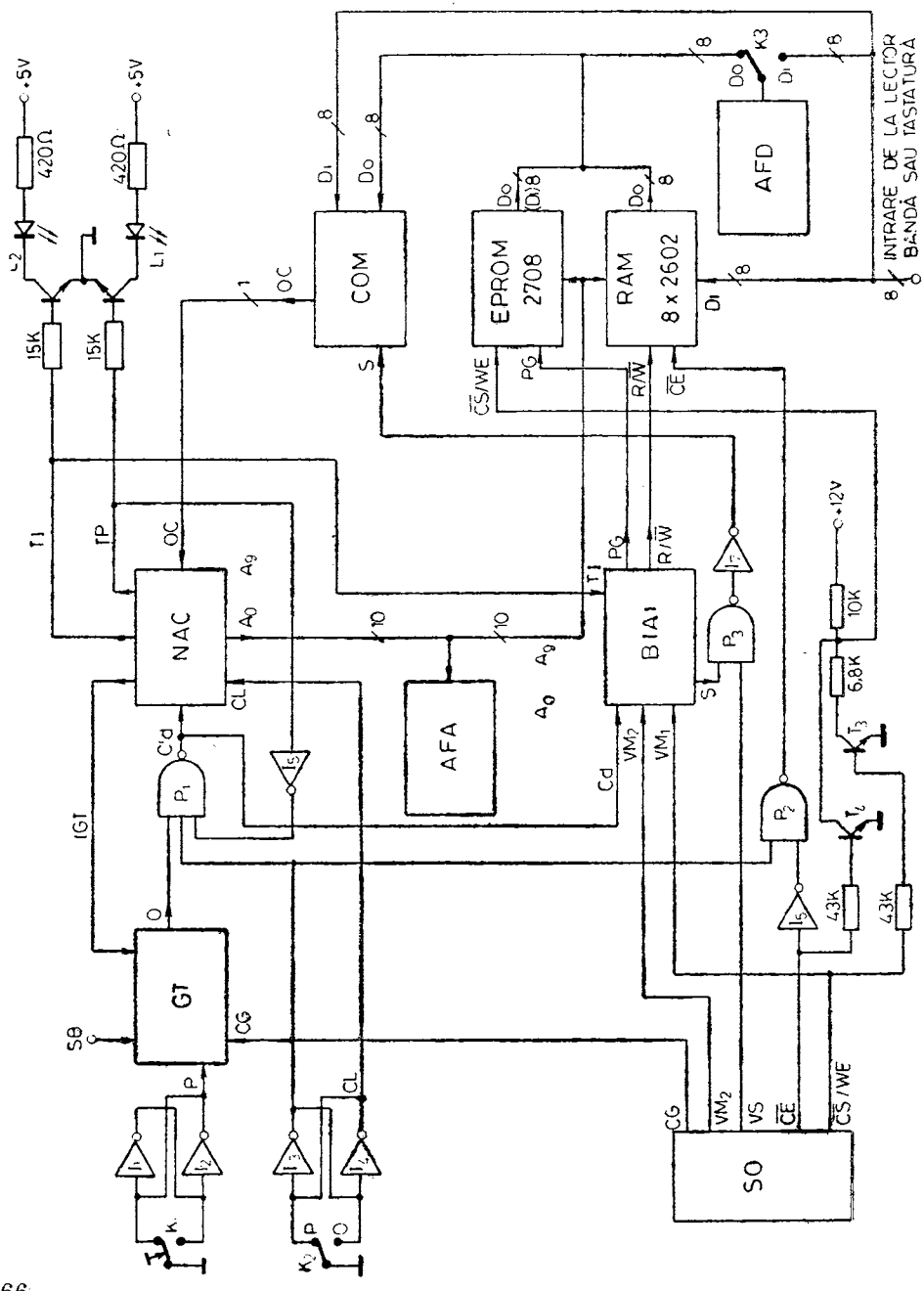


Fig. 2.29. Schema bloc a programatorului autonom

e. verificarea automată a conținutului memoriei EPROM prin comparare cu informația de pe banda perforată.

Alegerea operației dorite se realizează cu ajutorul selectorului de operații SO, constituit dintr-un comutator rotativ cu cinci poziții, și un codificator.

Funcționarea programatorului este comandată de către generatorul de tact *GT* care poate funcționa în regim declanșat (intrările *P* și *SB*) sau relaxat. Trecerea din regim relaxat în regim declanșat se face prin acțiunea semnalului logic *CG*, furnizat de selectorul de operații *SO*. Semnalul *IGT* duce la inhibarea funcționării *GT*, atunci cînd pe parcursul operației de verificare a conținutului memoriei EPROM prin comparare cu conținutul benzii perforate, se constată o neconcordanță. Funcționarea *GT* în regim de impulsuri singulare se obține prin acționarea manuală a comutatorului cu revenire K_1.

Numărătorul de adresă și de cicluri, NAC, asigură modificarea secvențială a adresei memoriilor RAM și EPROM și numără ciclurile de programare, semnalizînd prin semnalul *TP* și LED-ul L_1 terminarea operației de programare. Semnalul *TP* invalidează și poarta P_1 împiedicînd în continuare accesul impulsurilor de comandă la intrarea *NAC*. Semnalul *TI* și LED-ul L_2, corespund terminării operației de introducere a informației în RAM. Sennalul *TI*, prin intermediul blocului BIAI, întrerupe aplicarea impulsurilor de înscriere la intrarea $R/\overline{W}$ a memoriei RAM, după terminarea încărcării acesteia. Aducerea la zero a numărătorului NAC se face prin semnalul CL furnizat de către comutatorul „pornit"-„oprit", K_2.

Blocul de întîrziere și amplificator al impulsurilor de programare BIAI are rolul de a genera impulsurile de programare cu amplitudine de 26 V și durată de 0,8 msec, întîrziate cu 15 microsecunde față de fronturile de comandă ale NAC. În același timp el furnizează, în timpul operației de încărcare a RAM, semnalul de înscriere, cu o durată de 20 microsecunde, aplicat intrării $R/\overline{W}$ și semnalul de strob *S* pentru comparatorul COM care servește pentru verificarea concordanței conținutului memoriei EPROM programate și a benzii perforate. Semnalele WM_1 și VM_2, furnizate de selectorul de operații și aplicate BIAI, validează funcționarea a două monostabile conținute în acesta corespunzător cu operația în curs de efectuare.

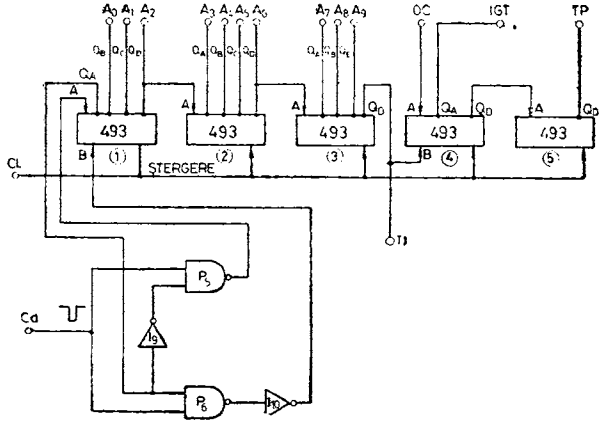


Fig. 2.30. Numărătorul de adresă și cicluri

Programatorul este prevăzut cu afișarea datelor (AFD), existînd posibilitatea comutării dispozitivului de afișaj, (8 LED-uri comandate cu tranzistoare) la intrarea de date a programatorului sau la ieșirea memoriilor RAM și EPROM. Este necesară și afișarea celor zece biți de adresă care se realizează prin AFA.

Circuitul prevăzut cu tranzistoarele T_3 și T_4 comandate de SO asigură aplicarea la intrarea $\overline{CS}/WE$ a memoriei EPROM, a potențialelor indicate în foaia de catalog în funcție de operația în curs de efectuare (+12 V, +5 V și OV).

În cele ce urmează se va prezenta structura principalelor blocuri componente ale programatorului, în ideea facilitării realizării acestuia de către cititorul interesat.

Numărătorul de adresă și cicluri este prezentat în figura 2.30. Sînt utilizate numărătoare binare de cîte patru biți de tipul 493. Primele trei capsule sînt utilizate în principal pentru realizarea numărătorului de adresă (10 biți). Bistabilul A, al primei capsule, are rol de celulă de memorare în circuitul format cu P_5, P_6, I_9 și I_{10}; circuit care asigură ca, după ștergerea numărătorului cu ajutorul semnalului CL, primul impuls de comandă aplicat la intrarea Cd să nu modifice conținutul numărătorului de adresă. O astfel de funcționare este necesară pentru asigurarea încărcării memoriei RAM, respectiv programării memoriei EPROM începînd cu adresa zero. Ultimul bistabil al celei de a treia capsule face parte din numărătorul de cicluri împreună cu alte șapte bistabile conținute în capsulele patru și cinci. Primul bistabil din capsula patru este comandat de la ieșirea blocului de comparare COM cu sennalul OC, iar la apariția unei neconcordanțe între conținutul memoriei EPROM programate și al benzii perforate furnizează semnalul IGT care invalidează funcționarea GT, comandat de la perforația de antrenare a benzii, oprind numărătorul de adresă la adresa la care s-a constatat neconcordanța. Numărătorul de cicluri asigură efectuarea a 128 de cicluri de programare după care semnalul TP invalidează poarta P_1, iar LED-ul L_2 semnalizează terminarea programării (figura 2.29). Semnalul de terminarea încărcării memoriei RAM, TI, interzice aplicarea în continuare a impulsurilor de înscriere la intrarea $R/\overline{W}$ a memoriei RAM, și acționează LED-ul de semnalizare a terminării încărcării L_1.

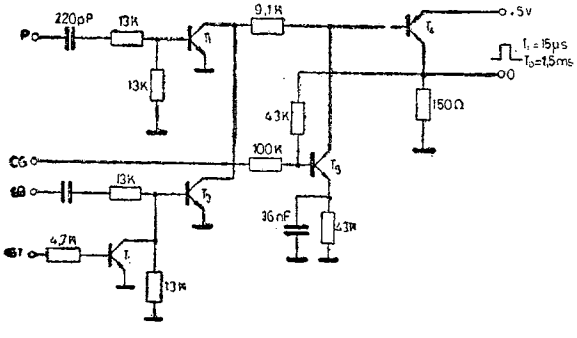


Fig. 2.31. Generatorul de tact

În figura 2.13 se prezintă generatorul de tact

constituit din generatorul autoblocat capacitiv (T_4, T_5), circuitele de declanşare (T_1, T_2) şi circuitul de interdicţie al declanşării prin intrarea SB (T_3). Pentru CG pe unu logic generatorul funcţionează relaxat cu perioada de repetiţie de 1,5 ms. Intrarea P este folosită în regim declanşat pentru acţionarea manuală (K_1) pas cu pas, iar intrarea SB primeşte impulsurile pozitive de sincronizare de la perforaţia de anternare a benzii sau de la tastatură.

Blocul de întîrziere şi amplificatorul impulsurilor de programare este redat în figura 2.32. Monostabilul M_1 formează impulsurile de programare cu o durată de 0,8 ms, iar M_2 impulsurile de înscriere în memoria RAM ($R/\overline{W}$) şi impulsurile de strob (S) pentru blocul de comparare COM. Amplificatorul cu tranzistoare asigură nivelul de 26 V al impulsurilor de programare şi nivel de ieşire nul în situaţiile cînd se efectuează alte operaţii decît programarea.

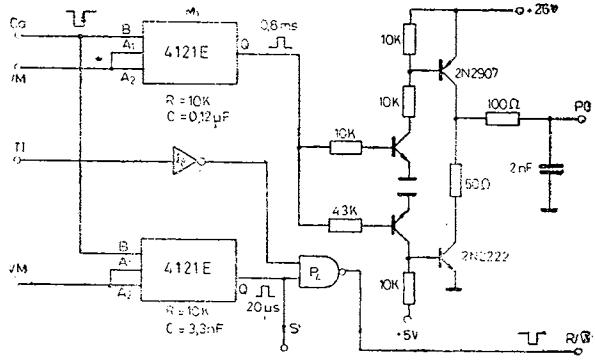


Fig. 2.32. Blocul de întîrziere şi de amplificarea impulsurilor de programare

În figura 2.33 se prezintă selectorul de operaţii compus din comutatorul cu cinci poziţii K şi decodificatorul cu tranzistoare. Numerotarea contactelor corespunde următoarelor operaţii : 1 — încărcare RAM, 2 — verificare RAM, 3 — programare, 4 — verificare EPROM manual şi 5 — verificare EPROM prin compararea cu banda perforată.

Comparatorul COM este realizat cu opt porţi sau exclusiv (două capsule 486) la ieşirea cărora este conectată cîte o intrare a opt porţi $NU—SI$ cu două intrări şi colector în gol (două capsule

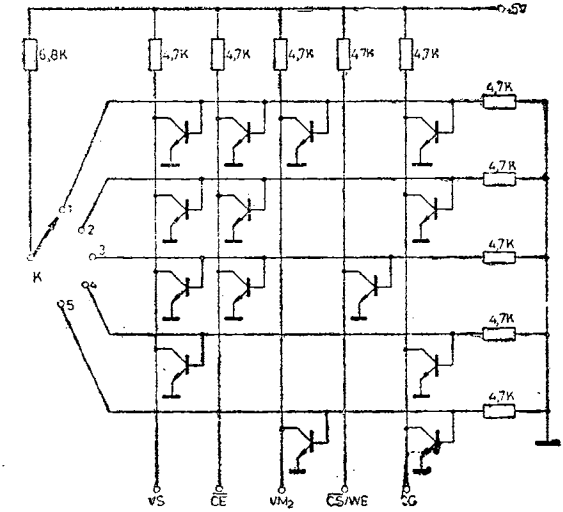


Fig. 2.33. Selectorul de operaţii

403). La cea de a doua intrare a tuturor porților *NU—SI* se aplică impulsul de strob *S*. Ieșirile tuturor porților sînt cablate la o singură rezistență de 2,2 kohmi, obținîndu-se ieșire OC.

Este de menționat că toate tranzistoarele utilizate, cu excepția celor notate sînt de tipurile BC 172 și BC 251.

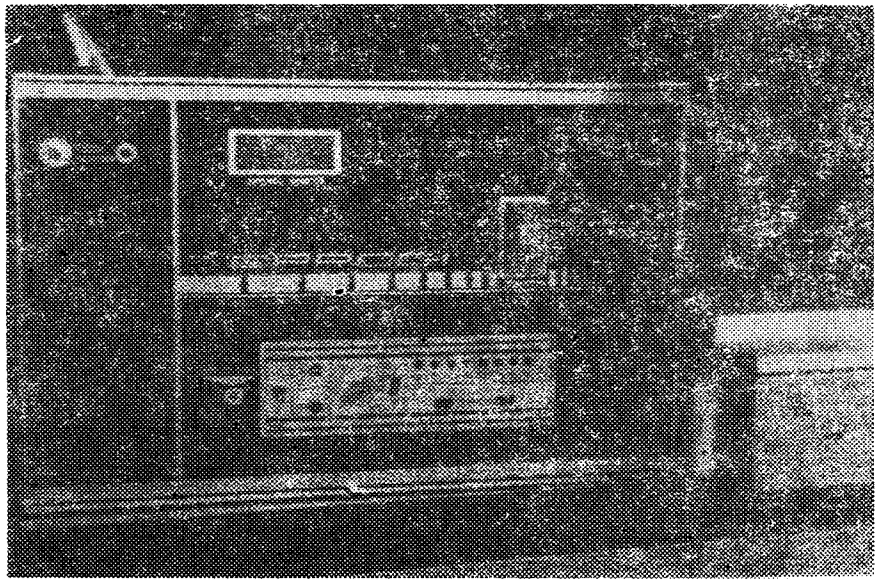


Foto 2.4. Vedere generală a programatorului autonom

În foto 2—4 este prezentat aspectul general al programatorului realizat la Catedra de electronică de la Facultatea de electrotehnică din Timișoara după concepția prezentată mai sus.

Referitor la utilizarea programatorului sînt de menționat următoarele: introducerea și scoaterea din soclu a memoriei EPROM se face cu sursele de alimentare deconectate; manipularea selectorului de operații se face numai cu comutatorul K_2 pe poziția 0 și declanșarea oricărei operații din cele cinci posibile se face prin trecerea comutatorului K_2 pe poziția *P*.

BIBLIOGRAFIE

1. * * * *8080 microcomputer systems user's manual*, 1975.
2. * * * *Documentația tehnică pentru miniimprimanta SEIKO Model 104.*
3. Al. Rogojan, C. Strugaru, M. Stratulat, *Interfață pentru imprimanta numerică*, Buletinul științific și tehnic al Institutului Politehnic „Traian Vuia", Timișoara, serie electrotehnică, Tom 24 (38), fasc. 1—1978.
4. J. D. Nicoud, *Alfanumeric TV display interface*, Microscope vol. 1. Nr. 6, 1976.
5. D. Del Corso, *A programmer for the 2708*, Microscope, vol. 2, Nr. 10, 1977.

CAPITOLUL 3

REALIZAREA UNUI MICROCALCULATOR: VARIANTA 2.

3.1. INTRODUCERE

Construit tot pe baza microprocesorului 8080, microcalculatorul descris în acest capitol, prezintă elemente structurale și aspecte funcționale noi care îl deosebesc de varianta precedentă.

Una din principalele deosebiri, după cum se va constata mai în detaliu în cuprinsul capitolului, o constituie absența panoului. În locul cheilor pentru formarea cuvintelor de adresc sau de date, a circuitelor de afișare, circuite pentru adresarea directă a memoriei, etc. care intră obișnuit în compunerea panoului, acesta a fost înlocuit cu o tastatură și un program înregistrat într-o memorie fixă de tip ROM care împreună cu microprocesorul, asigură toate funcțiunile panoului din cap. 2. plus alte funcțiuni care ușurează operarea microcalculatorului și punerea la punct a programelor.

Microcalculatorul posedă un sistem de întreruperi care permite efectuarea transferului cu echipamente periferice prin cereri de întrerupere.

În cuprinsul capitolului mai sînt tratate un număr de interfețe, altele decît cele deja prezentate, cu lectorul și perforatorul de bandă, unitate de casetă magnetică și o variantă de interfață cu dispozitivul de afișare pe ecran TV, DAF 1001, completînd în acest fel gama de echipamente periferice descrise.

Se ajunge astfel la un ansamblu care în mod obișnuit poartă numele de microsistem de dezvoltare (fig. 3.1). [1]

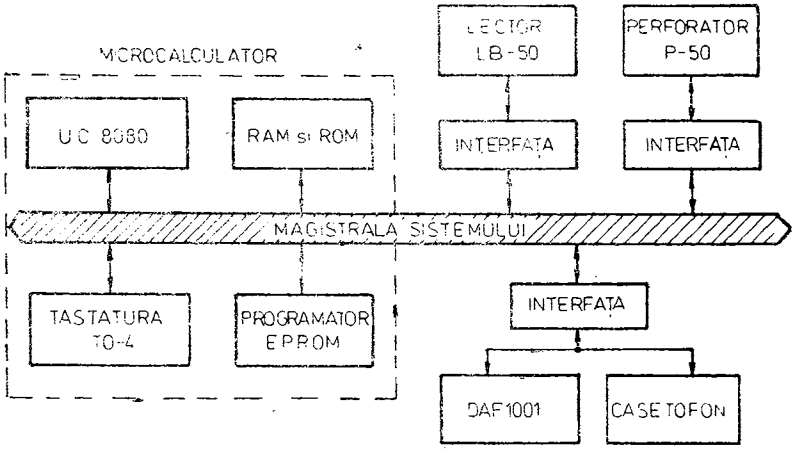


Fig. 3.1. Structura microsistemului de dezvoltare : varianta 2

În cazul programării în limbaj mașină acest sistem poate servi cu succes implementării unor aplicații a căror programe nu depășesc 500—1 000 de octeți de memorie ROM.

Dincolo de aceste dimensiuni ale programelor aplicative scrierea și depanarea în cod mașină devine costisitoare fără dotarea microsistemului cu programe de bază ca editor, asamblor, compilator, ș.a. [2].

Nu este însă neapărat necesar să se dispună de aceste programe deoarece se poate folosi un cross-asamblor și un interpretator implementate pe un calculator de capacitate mijlocie. [3].

Programele utilizator scrise în limbajul de asamblare al microprocesorului 8080 pot fi testate și puse la punct pe acest calculator care oferă în plus și toate facilitățile obișnuite ale unui sistem de calcul (de generația a treia). La capătul etapei de elaborare și verificare a programului acesta va fi de preferință extras sub formă de program obiect pe bandă perforată.

După încărcarea programului obiect de pe bandă perforată în memoria microsistemului de dezvoltare se verifică funcționarea de ansamblu și dacă sînt satisfăcuți toți parametrii propuși se poate trece în cele din urmă la înregistrarea programului într-o memorie PROM.

Utilizarea microsistemului de dezvoltare în cadrul unor aplicații de robotică este arătată în capitolul 7.

3.2. MICROCALCULATORUL DE DEZVOLTARE

3.2.1. CONCEPȚIA DE BAZĂ

Circuitul LSI microprocesor poate fi considerat o componentă electronică avînd un repertoriu de funcțiuni logice care se realizează pe baza proprietăților combinatorii și secvențiale ale acestuia la care se adaugă caracterul programabil. Împreună cu alte componente programabile microprocesorul permite proiectarea unor scheme logice secvențiale complexe pe baza conceptului proiectării logice programate.

Aplicat la structura microcalculatorului de dezvoltare, conceptul proiectării logice programate se traduce prin existența unei memorii fixe de programe care permite implementarea funcțiunilor de bază ale panoului de comandă și altele mai complexe (fig. 3.2).

Să examinăm mai îndeaproape la ce mod de funcționare conduce programul aflat în memoria fixă ROM. Principial diferit de panoul de comandă descris în subcapitolul 2.6., care pentru a obține accesul direct la memoria RAM (DMA) pune microprocesorul în strea inactivă *HOLD*, programul fix permite microprocesorului să fie în permanență activ. Acest lucru este de natură să faciliteze implementarea unor funcții dorite de utilizator cu o importantă economie de circuite.

Programul care reunește funcțiunile de bază ale microcalculatorului din punct de vedere al utilizatorului este în mod obișnuit numit MONITOR [4].

Activitatea microprocesorului în conformitate cu instrucțiunile programului MONITOR este ilustrată în organigrama simplificată din figura 3.3.

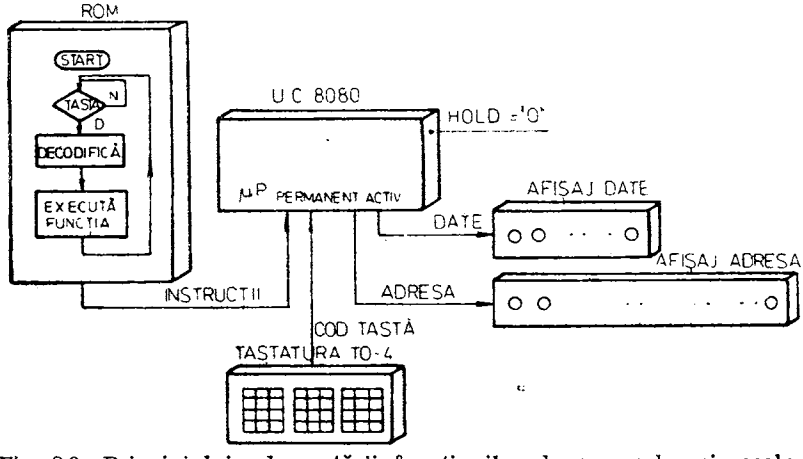


Fig. 3.2. Principiul implementării funcțiunilor de panou la microcalculatorul de dezvoltare

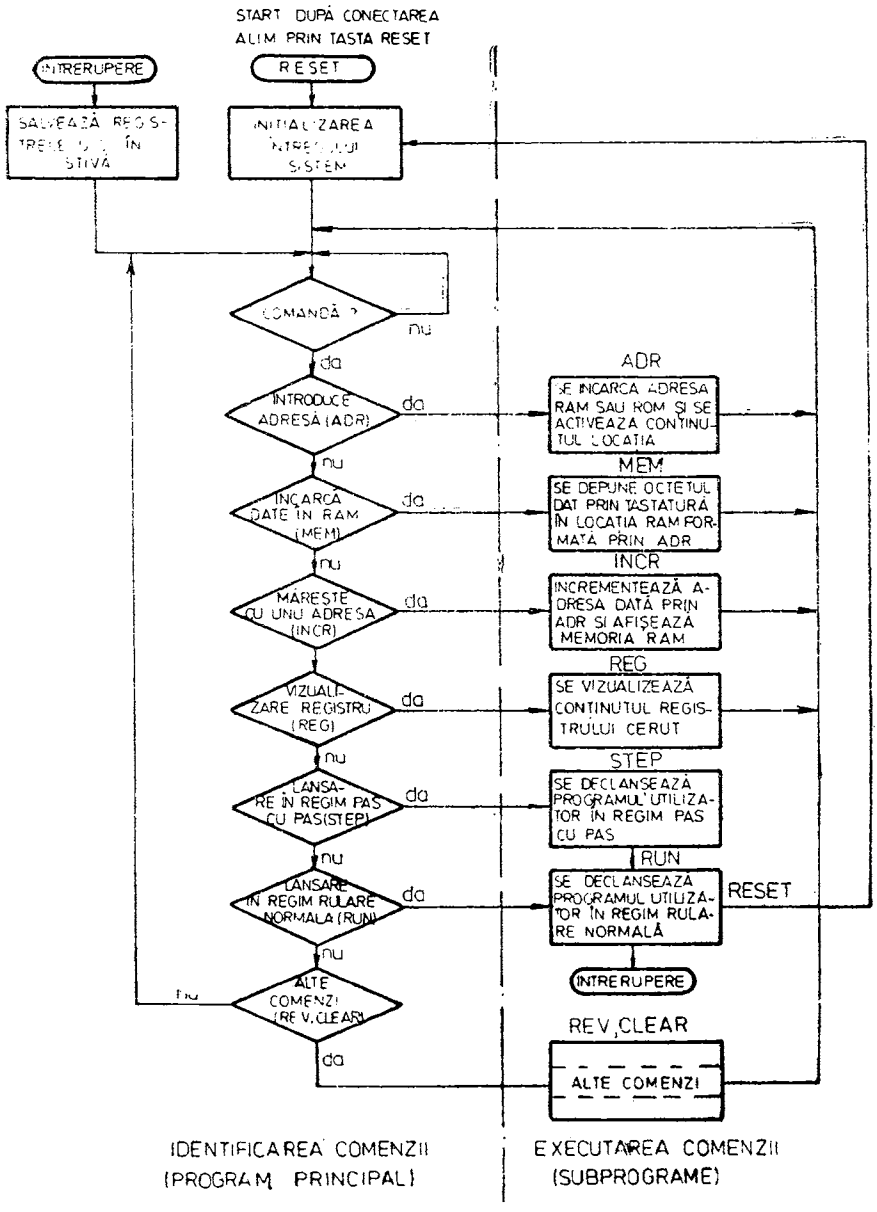


Fig. 3.3. Organigrama simplificată a programului MONITOR

Tastatura este inspectată continuu pentru sesizarea tastei acționate. Dacă utilizatorul apasă o tastă codul generat de electronica de explorare a tastaturii este introdus în microprocesor și decodificat prin procedee specifice tehnicii programării. După identificarea funcției cerute se execută saltul la subprogramul care efectuează operațiile solicitate prin tasta acționată.

Tasta *ADR* permite de exemplu, introducerea unei adrese de memorie și citirea locației respective precum și adresa de început a unui program. Apăsarea tastei *ADR* are drept consecință apelarea subprogramului *ADR*, în care se așteaptă transmiterea consecutivă a patru cifre hexazecimale ce formează adresa. După transmiterea ultimei cifre hex are loc asamblarea cifrelor hex în perechea de registre H, L. Indiferent dacă adresa introdusă de utilizator reprezintă o adresă de memorie în care se intenționează să se scrie sau să se citească sau reprezintă adresa de start a unui program, rezultatul va fi afișarea pe diodele luminiscente a adresei dorite și a octetului de date corespunzător locației de memorie adresate.

Prin urmare comanda *ADR* $XXXX_{16}$ oferă posibilitatea citirii unei locații de memorie și mai ales a lansării unui program utilizator de la oricare adresă, aceasta spre deosebire de modul de lansare prin panoul de comandă prezentat în cap. 2, la care adresa de start este întotdeauna fixă, 0000_{16}.

Asemănător sînt implementate alte comenzi avantajoase printre care cele de vizualizare a registrelor interne ale microprocesorului.

O notă aparte o au acele comenzi pentru care în afară de tratarea prin program (sau fără tratare deloc) este necesară generarea unor impulsuri speciale care să fie recepționate la nivelul magistralei de comandă a microcalculatorului. Astfel de comenzi sînt inițializarea (*RESET*), rulare normală (*RUN*) sau pas cu pas (*STEP* și întreruperea de operator (*INTROP*).

3.2.2. CONFIGURAȚIA MICROCALCULATORULUI

Schema bloc a microcalculatorului de dezvoltare este dată în figura 3.4. Se remarcă în primul rînd grupul circuitelor unitate centrală de microprocesor 8080 și generatorul de tact 8224 caracteristice sistemelor bazate pe acest tip de microprocesor. Octetul de stare al microprocesorului este memorat într-un registru al cărui conținut este afișat în permanență de un grup de opt diode luminiscente.

Capacitatea memoriei este de 1 Koctet RAM și maximum 3 Kocteți de memorie fixă EPROM.

Generarea semnalelor de comandă ale microcalculatorului se realizează în blocul cu acest nume implementat cu circuite integrate TTL.

Pentru urmărirea activității microcalculatorului în regimul de rulare pas cu pas, în afara octetului de stare deja amintit, sînt prevăzute afișări cu diode luminiscente pentru cei 8 biți de date și 16 de adresă.

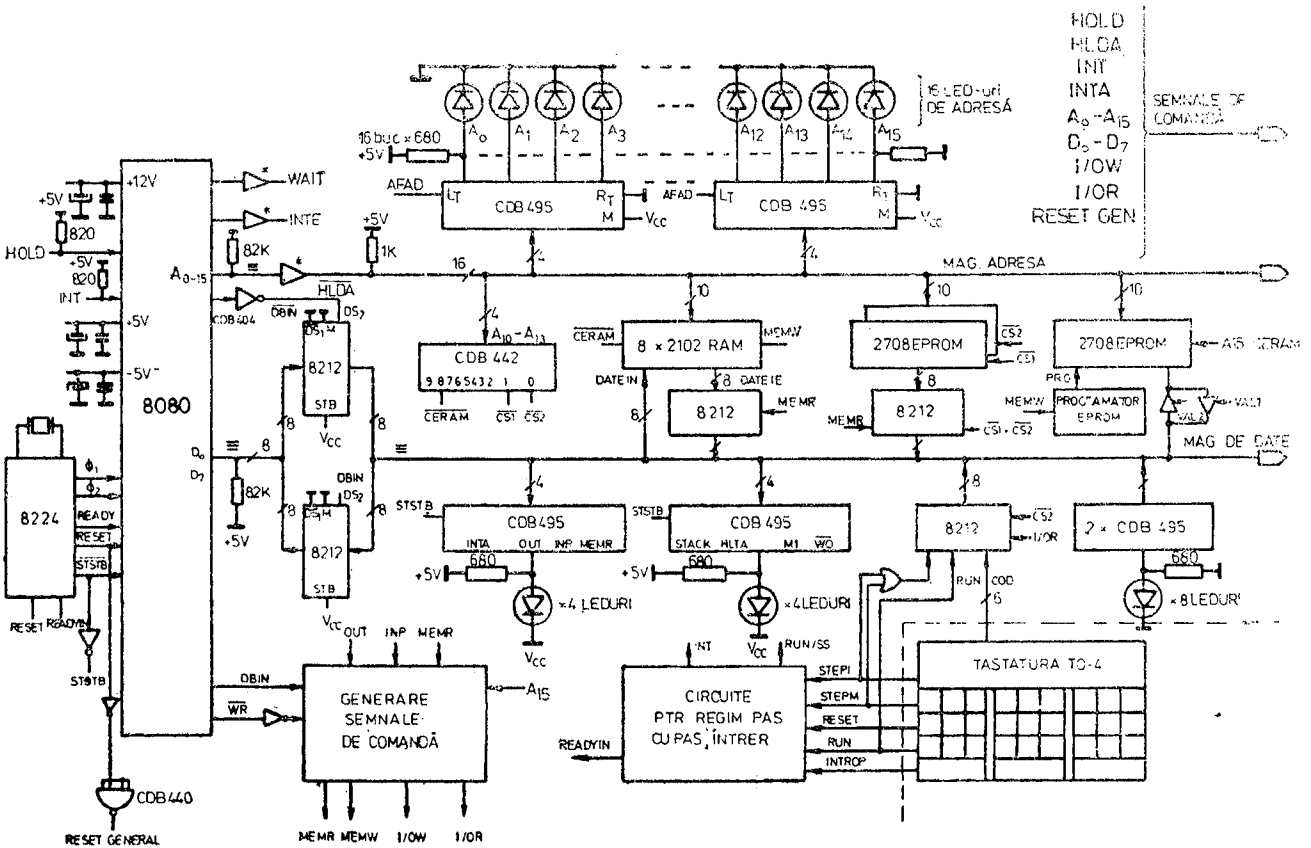


Fig. 3.4. Schema bloc a microcalculatorului de dezvoltare

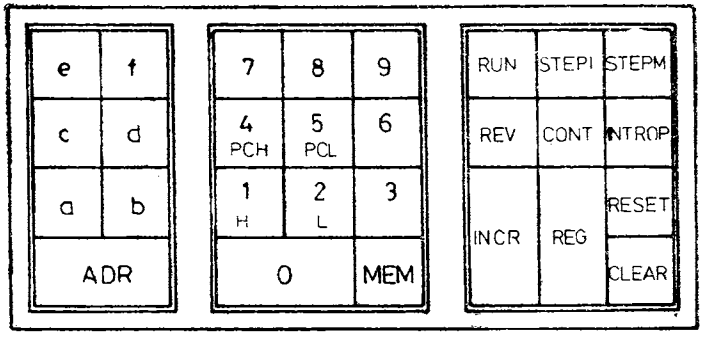


Fig. 3.5. Comenzile microcalculatorului

Introducerea datelor și darea comenzilor se face prin intermediul tastaturii operative *TO*-4 care dispune de electronică de explorare proprie. În contextul variantei de microcalculator din cap. 2 tastatura *T* 0-4 era folosită doar pentru introducerea datelor în hexazecimal. În schema de față din cele 28 de taste ale lui *TO*-4, 12 au fost afectate comenzilor (vezi funcțiile acestora în tab. 3.1) restul reprezentînd cele 16 cifre hexazecimale (fig. 3.5).

Tabelul 3.1.

Lista comenzilor și funcțiile realizate

Numele comenzii	Descrierea funcțiunilor
REST	Inițializează întregul sistem și aduce la zero numărătorul de adrese PC
ADR	Permite introducerea unei adrese în hexazecimal și afișează conținutul memoriei
MEM	Depune un octet în memorie și îl afișează
REG	Vizualizează registrul intern după declanșarea întreruperii INTROP
REV	Revenirea la programul utilizator după întrerupere
STEPI	Regim pas cu pas, instrucție cu instrucție
STEPM	Regim pas cu pas, ciclu cu ciclu
RUN	Declanșarea regimului de rulare automată a programului utilizator
CLEAR	Anulează ultima tastă apăsată
CONT	Comanda de preluare
INCR	Mărește cu unu conținutul adresei aflate în HL
INTROP	Declanșează o întrerupere în orice moment indiferent de regimul de lucru

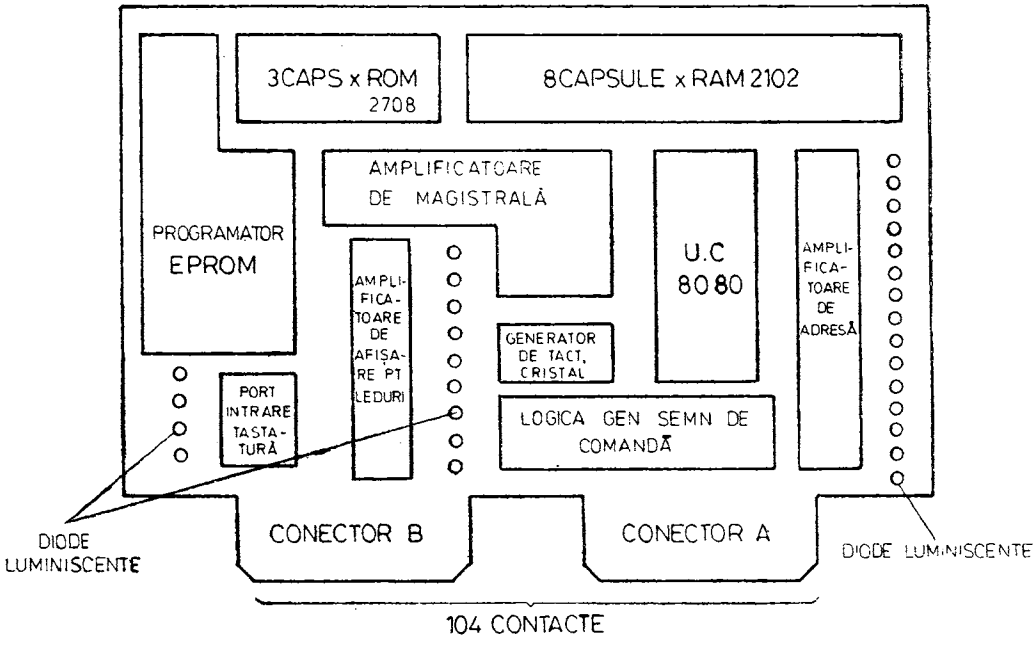


Fig. 3.6. Amplasarea blocurilor microcalculatorului pe placa de circuit imprimat.

Tastatura este conectată atît la nivelul magistralei de date, pentru transmiterea spre microprocesor a codului tastei, cît și la nivelul magistralei semnalelor de comandă și blocului de stabilire a regimurilor de lucru.

În fine configurația este completată de un programator de memorii EPROM avînd o schemă identică cu cea prezentată în cap. 2.

Întreaga schemă din figura 3.4, cu excepția tastaturii *TO*-4, este realizată pe o singură placă de circuit imprimat, ceea ce constituie o trăsătură a microcalculatorului de dezvoltare (fig. 3.6).

Descrierea în continuare a microcalculatorului cuprinde elemente funcționale și de detaliu ale părților mai importante ale schemei, prezentarea succintă a programului MONITOR, modul de operare și probleme ale punerii la punct.

3.2.3. CIRCUITELE DE MEMORIE

Modul în care sînt conectate memoriile RAM și ROM trebuie să asigure transferul datelor între microprocesor și circuitele menționate în ambele sensuri pentru RAM și într-un singur sens pentru ROM.

Selecția celor 8 capsule 2102 (1K×1 bit) se face cu semnalul CERAM generat prin decodificarea rangurilor A_{10}, A_{11}, A_{12} și A_{13} ale adresei. Ceilalți biți A_9-A_0 constituie adresa unui octet în memorie, iar biții A_{15} și A_{14} au destinație specială cum se va vedea ulterior. Ecuația logică a semnalului *CERAM* este $CERAM=\bar{A}_{13}.\bar{A}_{12}.A_{11}.\bar{A}_{10}$. Liniile de date D_0-D_7 sînt legate direct la intrările de date ale capsulelor de memorie dar ieșirile acestora sînt trecute printr-un circuit amplificator trei stări 8212. Acest lucru este de natură să permită ieșirilor memoriilor RAM să aibă un factor de încărcare suficient pentru a comanda liniile de date în cazul unui transfer între memorie și circuitele periferice (acces direct la memorie DMA)

Timpul acces al memoriilor RAM 2102 (350 ns) fiind mai scurt decît durata stării T_3 (500 ns) la capătul căreia datele trebuie să fie deja scrise sau citite, nu mai sînt necesare măsuri de sincronizare prin intermediul semnalului *READY*.

Memoria de programe este formată din două capsule EPROM 2708 (1K×8 bit) adică 2 Kocteți. Dacă este nevoie se mai poate adăuga încă 1 Koctet, utilizînd capsula EPROM, care este conectată la schema programatorului de memorii.

Primele două capsule sînt selectate independent de către semnalele $CS1=\bar{A}_{13}.\bar{A}_{12}.\bar{A}_{11}.A_{10}$ și $CS2=\bar{A}_{13}.\bar{A}_{12}.\bar{A}_{11}.\bar{A}_{10}$. Adresarea unui octet în interiorul fiecărei capsule se face de către liniile de adresă A_0-A_9. Aceeași soluție de mărire factorului de încărcare la ieșirea memoriilor EPROM a fost luată prin introducerea unui circuit 8212 conectat ca amplificator trei stări. Timpul de acces al unei memorii EPROM 2708 fiind de 400 ns nu e nevoie de sincronizare cu microprocesorul.

De remarcat că semnalul *CS2* este validat de o combinație a liniilor de adresă $A_{13}\div A_{10}$ avînd nivelul 0 logic. Prima locație a acestei capsule are adresa $A_9\div A_0=0$. Prin urmare adresa cu valoarea 0000_{16} (biții A_{15} și A_{14} avînd alt rol) este adresa primului octet al acestei capsule. Deoarece imediat după inițializarea prin *RESET*, numărătorul de adrese (PC) al microprocesorului conține zero peste tot devine clar de ce programul MONITOR a fost înscris în această capsulă.

Faptul că se citește din memoria RAM sau din ROM înseamnă validarea unuia sau a celuilalt din circuitele amplificatoare 8212. De aceea selecția lor se face independent printr-un semnal comun READM care indică transfer cu memoria în general și două semnale diferite CERAM și CS1+CS2.

3.2.4. MAGISTRALELE DE ADRESĂ ȘI DE DATE

Cele 16 linii unidirecționale ale semnalelor magistralei de adrese sînt amplificate de circuite neinversoare cu colector în gol CDB 407. Deși aceste circuite au un timp de comutație mai lent decît al amplificatoarelor trei stări, nu apar

probleme, deoarece liniile de adresă comută o singură dată într-un ciclu de mașină, anume în prima stare T_1, iar datele sînt citite (scrise) abia în cursul stării T_3.

Rezistențele de colector ale circuitelor 407 au fost dimensionate la valoarea 1 Kohm. Astfel se pot comanda pînă la 21 sarcini *TTL*.

Datorită circuitelor cu colector în gol este posibilă realizarea funcției SAU CABLAT pe magistrala de adresă de care e nevoie în transferul de date prin acces direct la memorie (*DMA*). Pentru ca funcția SAU să se execute corect, perifericul care cere accesul direct la memorie trebuie să aducă microprocesorul în starea *HOLD*. În această stare liniile de adresă devin flotante. Totuși la intrarea amplificatoarelor semnalelor de magistrală va apare un nivel logic corect 1, datorită rezistențelor legate spre +5 V. Prin urmare la ieșirea amplificatoarelor se va putea instala orice nivel logic cerut de periferic.

Realizarea magistralei de date pune probleme mai deosebite din două motive. Pe această magistrală microprocesorul transmite sau recepționează date liniile sînt deci bidirecționale și trebuie să se asigure o multiplexare în timp. Pe de altă parte liniile de date comută și de patru ori într-un ciclu de mașină și diagramele de timp devin mai pretențioase.

Soluția care se impune este folosirea a două grupuri de 8 amplificatoare rapide cîte un grup pentru fiecare direcție avînd intrări de autorizare separate (de grup). În plus amplificatoarele vor fi de tipul trei stări dacă se dorește menținerea în continuare a magistralei cu trei stări. Este mai avantajoasă menținerea unei magistrale de date cu trei stări, deci bidirecțională și după amplificare, în dauna variantei unor magistrale de date separate de intrare și ieșire, pentru că numărul de trasee conductoare pe cablajul imprimat spre alte module este păstrat minim. Această problemă „separat sau trei stări" se mai pune odată la scoaterea liniilor de date la conector pentru a fi legate la sistem. În cazul de față s-a ales din nou soluția trei stări dar din altă rațiune, anume preferința pentru semnale de comandă, punctele de conexiune fiind limitate ca număr.

Revenind la amplificatoarele semnalelor magistralei de date, o pereche de circuite 8212 conectate ca porți trei stări și validate unul de nivelul 1 logic și celălalt de nivelul 0 logic al semnalului *DBIN*, rezolvă problema.

În tab. 3.2 se arată cum au fost repartizați biții de adresă A_0—A_{15} pentru selectarea circuitelor RAM, ROM a echipamentelor periferice și ale regimurilor rulare normală respectiv pas cu pas.

3.2.5. ORGANIZAREA INTRĂRILOR ȘI IEȘIRILOR

Este deja cunoscută importanța generării semnalelor de comandă care asigură citirea/scrierea datelor în memorie sau introducerea/extragerea datelor schimbate cu perifericele.

Tabelul 3.2.

Adrese de selecţie

MEMORIE	ADRESA
ROM 1	00 00
ROM 2	04 00
PROM	A0 00
RAM	20 00
PERIFERICE	**ADRESA**
PORT TASTATURA	80 00
SENS MOTOR (1—8)	80 00
SENS MOTOR (9—12)	84 00
TACT MOTOR (1—8)	A8 00
TACT MOTOR (9—12)	AC 00
COMUTATOARE (1—8)	84 00
PORT PERFORATOR	94 00
CP-PERFORATOR	98 00
NPACT-PERFORATOR	9C 00
LOG-PERFORATOR	9C 00
PORT LECTOR	8C 00
AVANS LECTOR	90 00
SENS LECTOR	8C 00
ÎNTRERUPERE	90 00
USART	A4 00
CMOD USART	A4 01
CONVERTOR A/N	88 00
SELECTOR CANALE	84 00

S-a arătat în capitolul 2 modul în care se pot obţine aceste semnale pe baza impulsurilor *DBIN* şi $\overline{WR}$ furnizate de microprocesor şi a decodificării octetului de stare după memorarea lui prealabilă.

Reluarea acestor probleme se va face dintr-o perspectivă nouă impusă de necesităţile microcalculatorului de dezvoltare.

Firma INTEL recomandă utilizarea semnalelor $\overline{MEMR}$, $\overline{MEMW}$, $\overline{I/OR}$ şi $\overline{I/OW}$ semnale produse de circuitul 8228 special destinat pentru generarea acestor semnale. Ele corespund organizării intrărilor-ieşirilor după metoda *I/I izolate* (fig. 3.7). Această metodă pune la dispoziţia utilizatorului un spaţiu de memorie de 64 Kocteţi (RAM sau ROM) şi 256 de dispozitive de intrare-ieşire. Transferul de date cu acestea din urmă se va efectua prin intermediul instrucţiilor IN *m* sau *OUT m*, unde *m* este numărul dispozitivului de interfaţă cu perifericul.

O altă metodă *I/I în spaţiul memoriei* (fig. 3.8) este numită aşa pentru că se renunţă la semnalele de strob $\overline{I/OR}$ şi $\overline{I/OW}$ (şi implicit la instrucţiile IN

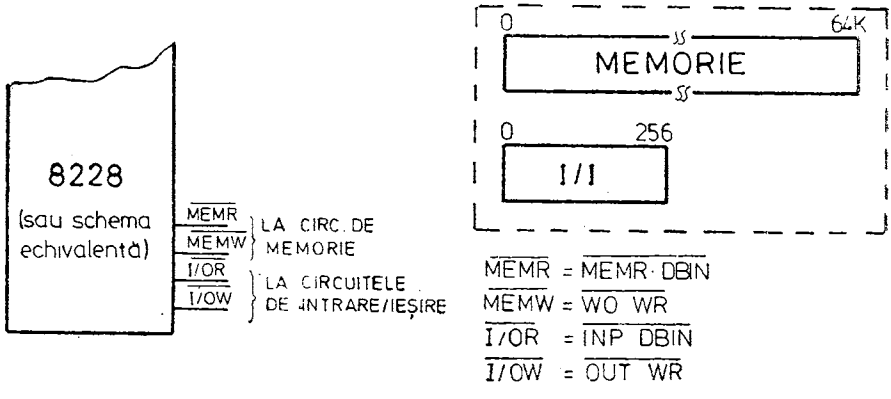


Fig. 3.7. Organizarea intrărilor-ieşirilor prin metoda I/I izolate

și OUT) și se reținn doar semnalele $\overline{MEMR}$ și; $\overline{MEMW}$. Distincția între circuitele de memorie și cele de intrare se face pe baza unui bit (sau mai mulți eventual) aparținînd liniilor de adresă, de exemplu A_{15}.

Pe această cale dispozitivele de intrare-ieșire pot fi adresate ca orice locație de memorie cu întreaga gamă de instrucții cu referire la memorie ceea ce reprezintă un avantaj cert din punct de vedere al programării [5].

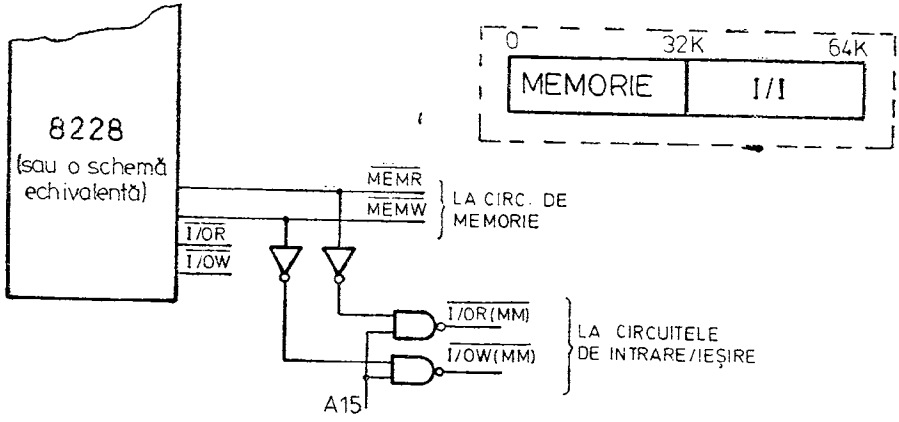


Fig. 3.8. Organizarea intrărilor-ieșirilor prin metoda I/I în spațiul memoriei

Microcalculatorul de dezvoltare folosește o a treia metodă de organizare reprezentînd o combinație între cele două metode menționate. Schema logică și ecuațiile semnalelor așa cum sînt ele obținute în blocul de GENERARE SEMNALE COMANDA sînt redate în fig. 3.9.

Acest mod de organizare a intrărilor-ieșirilor pune la dispoziția utilizatorului o flexibilitate sporită în proiectarea și experimentarea interfețelor cu diferite periferice.

3.2.6. AFIȘAREA ADRESEI, DATELOR ȘI OCTETULUI DE STARE

În scopul urmăririi funcționării microprocesorului în regimul de rulare pas cu pas și pentru vizualizarea memoriei, microcalculatorul este prevăzut cu 16 diode luminiscente pentru afișarea adresei, 8 pentru afișarea datelor și alte 8 pentru afișarea octetului de stare. Diodele luminiscente sînt montate pe placa de circuit imprimat a microcalculatorului. Comanda lor se face de la ieșirile registrelor CDB 495 cu rol de memorie tampon pentru nivelele logice care trebuie afișate.

Încărcarea registrelor destinate afișării adresei și datelor are loc pe frontul căzător al semnalului $AFAD=\overline{(STACK}.WR+WAIT).\emptyset_2$. Interpretarea aces-

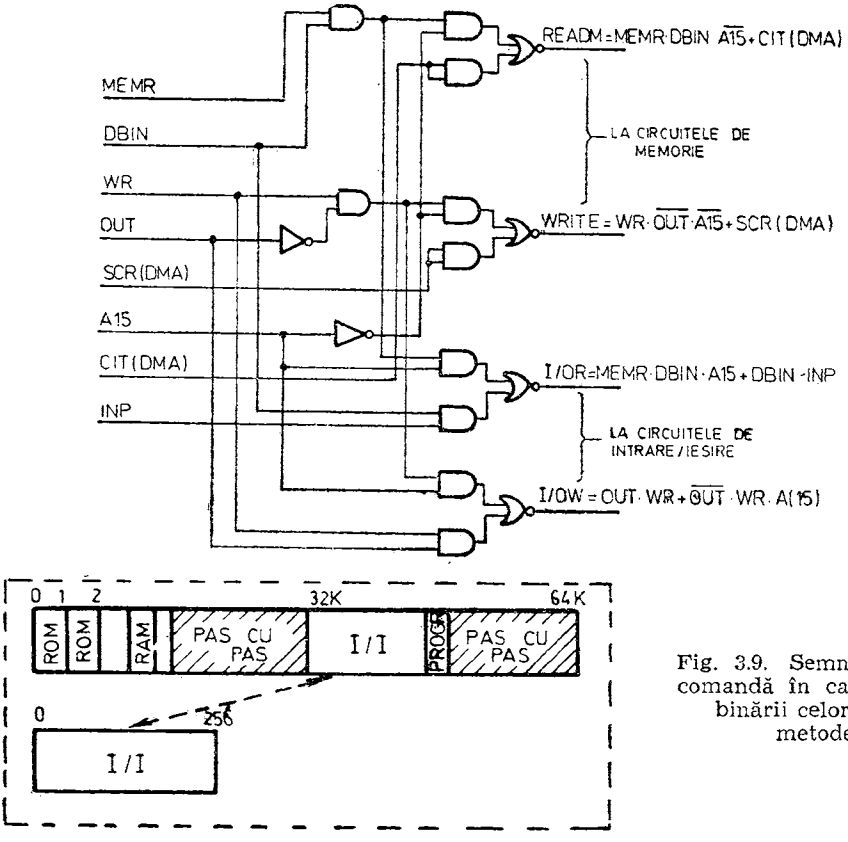


Fig. 3.9. Semnalele de comandă în cazul combinării celor două metode

tei ecuații arată că afișarea se produce sincron cu impulsul de tact $\emptyset_2$ pentru orice scriere în RAM sau periferic cu excepția stivei. Afișarea este repetată în cursul stării WAIT adică se împrospătează continuu conținutul registrelor de afișare în ritmul tactului $\emptyset_2$. În ce privește octetul de stare se reamintește că semnalul *STSTB* strobează informația de stare aflată pe liniile de date la începutul fiecărui ciclu de mașină și o memorează în registrul de afișare a stării.

Se atrage atenția asupra faptului că există numeroase alte variante de realizare a afișării (împrospătare continuă prin program, împrospătare prin acces direct la memorie, utilizarea unor module de 7 segmente cu memorie inclusă etc.) dar cea descrisă este printre cele mai economice.

Comanda prin program a schemei, pentru afișarea unui octet de date și a adresei, se poate face de către oricare din instrucțiile de scriere în memorie. Așa se prezintă, de exemplu, cazul vizualizării unei locații de memorie prin

comanda *ADR*, cînd, după introducerea adresei care indică locaţia de memorie, programul MONITOR citeşte octetul în acumulator şi apoi îl reînscrie în memorie, moment în care acesta apare pe diodele luminescente.

3.2.7. REGIMURI DE FUNCŢIONARE

Schema logică pentru stabilirea regimurilor de funcţionare (fig. 3.10) cuprinde patru bistabile *BSTEPI, BSTEPM, BRUN* şi *BINT*. Ultimul nu

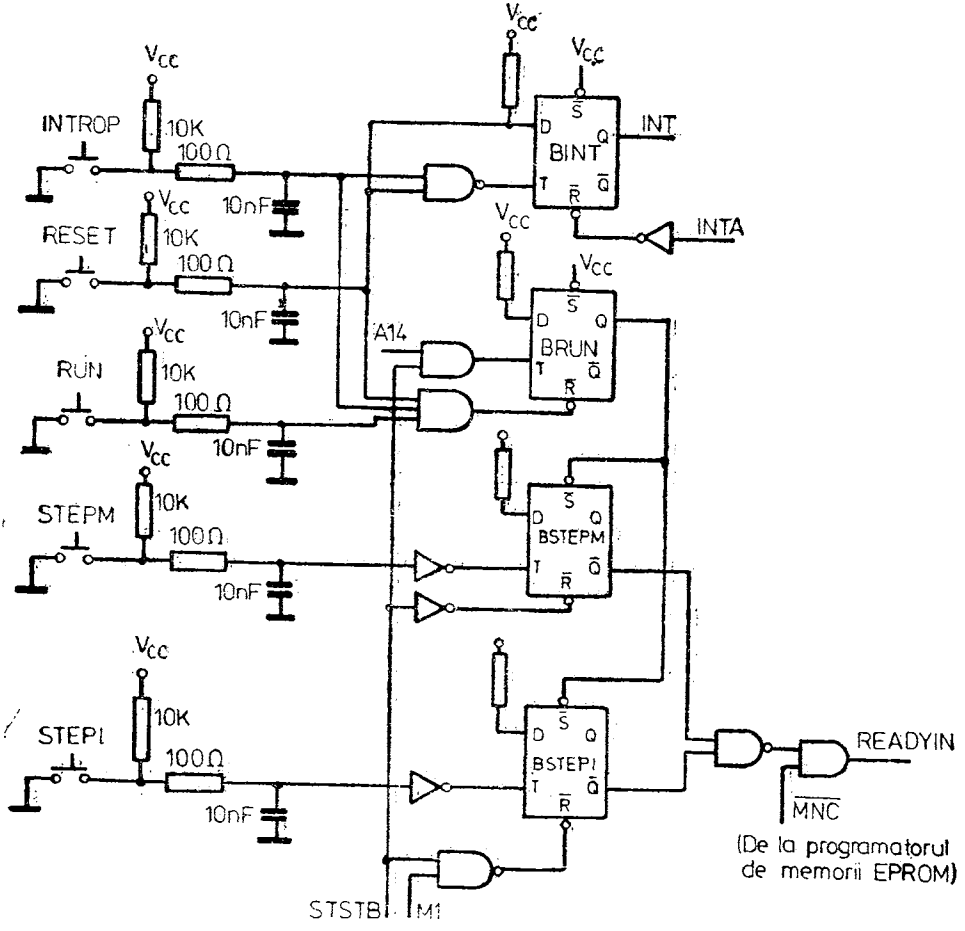


Fig. 3.10. Schema logică a blocului de stabilire a regimurilor de funcţionare

contribuie la instalarea nici unui regim ci memorează întreruperea de operator *INTROP*. În schimb primele trei colaborează la aducerea microcalculatorului în unul din regimurile rulare normală (*RUN*), rulare instrucție cu instrucție (*STEPI*), rulare ciclu cu ciclu (*STEPM*), prin combinații ale stărilor așa cum se arată în tabelul 3.3.

Tabelul 3.3.

Stabilirea regimurilor în funcție de bistabilele BRUN, BSTEPI, BSTEPM

BRUN	BSTEPI	BSTEPM	REGIMUL STABILIT
0	1	1	RULARE NORMALĂ
1	0	1	RULARE CICLU CU CICLU
	1	0	RULARE INSTRUCȚIE CU INSTRUCȚIE

Un număr de impulsuri venind de la tastatura TO-4 acționează intrările bistabilelor. Apăsînd tasta *RESET* se trimite impulsul cu acest nume spre bistabilele *BRUN* și *BINT*. Efectul este aducerea în starea 0 logic. Semnalul RESET este folosit și pentru inițializarea microprocesorului fiind conectat la intrarea *RESETIN* a circuitului 8224. Punerea pe 0 logic a bistabilului *BRUN* mai este asigurată de semnalul *RUN* generat de tasta cu acest nume. Întotdeauna poziționarea lui *BRUN* pe 0 logic implică forțarea stării 1 în bistabilele *BSTEPI* și *BSTEPM*.

Cele două semnale de ieșire ale schemei, *READYIN* și *INT*, sînt conduse la intrările respective ale circuitelor 8224 și 8080. Cu ajutorul lui *READYIN* se provoacă intrarea microprocesorului în starea de așteptare T_W, prin care se realizează oprirea microprocesorului între doi pași (ciclu de instrucție sau ciclu de memorie). Declanșarea unui pas se face prin tasta *STEPI* sau *STEPM*. Corespunzător se înscrie unul sau celălalt din bistabilele *STEPI* sau *STEPM*. Ecuația lui *READYIN* fiind, $READYIN = STEPI + STEPM$ microprocesorul iese din starea T_W și începe să parcurgă stările unui ciclu de instrucție sau ciclu de memorie. Oprirea microprocesorului după un pas se petrece în momentul apariției semnalului $STSTB.M1 = 1$, în cazul regimului instrucție cu instrucție, iar în cazul regimului ciclu de memorie, prin $STSTB = 1$. Aceste semnale aduc la zero bistabilele *BSTEPI* și *BSTEPM*.

Un avantaj clar al acestei scheme este posibilitatea rulării unui program, alternînd ciclurile de instrucție sau de memorie după dorință, prin simpla apăsare a tastelor *STEPI* sau *STEPM*. Pentru a realiza același lucru cu panoul de comandă prezentat la cap. 2 trebuie efectuată manevra de comutare a cheilor de panou *STEPI* și *STEPM* care e mult mai greoaie.

O problemă aparte o constituie trecerea de la regimul de rulare normală la regimul pas cu pas. Particularitatea stă în modul în care se realizează supra-

punerea operației de lansare a programului utilizator, care se face de către programul MONITOR, și aducerea microprocesorului în starea T_W.

Sincronizarea delicată a celor două procese se bazează pe utilizarea bitului A_{14} al adresei care, se subliniază, nu a primit nici o destinație pînă în acest moment.

Se presupune acum că adresa de lansare a programului utilizator se află deja încărcată în registrele *H*, *L* ca rezultat al acționării unei comenzi *ADR*. Lansarea în regim pas cu pas a programului utilizator se poate realiza cu oricare din tastele *STEPI* sau *STEPM*. Acestei acțiuni îi corespund două efecte (fig. 3.11). Un prim efect constă în încercarea de punere pe 1 logic a unuia din bistabilele *BSTEPI* sau *BSTEPM*, dar aceasta nu-și are rostul, deoarece bistabilul *BRUN* fiind pe 0 logic, ele sînt forțate în starea 1. Al doilea efect se petrece la nivelul programului MONITOR. Sesizînd apăsarea tastei de trecere în regimul pas cu pas MONITORUL execută un salt la subprogramul de lansare a unui program utilizator în regim pas cu pas. În cadrul acestuia se pune bitul H_6 din registrul H pe 1 logic și apoi o instrucțiune PCHL schimbă conținuturile registrelor PC și HL între ele.

După executarea acestei instrucțiuni următoarea adresă de program trimisă de microprocesor pe liniile de adresă va conține $A_{14}=1$. Intrarea de tact a bistabilului *BRUN* fiind sensibilă la frontul crescător al impulsului A_{14}. *STSTB* consecința va fi punerea imediată a bistabilului *BRUN* pe 1 logic. Astfel bistabilul *BSTEPI* și *BSTEPM* nu mai sînt forțate pe starea 1 și sub acțiunea lui *STSTB* ajung în starea 0 logic. Microprocesorul va intra prin urmare în starea de așteptare T_w gata să execute primul ciclu de mașină sau prima instrucțiune.

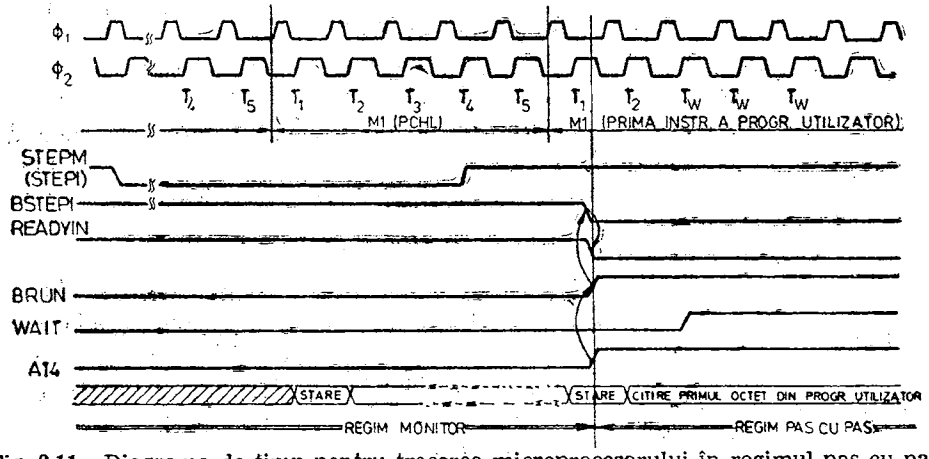


Fig. 3.11., Diagrama de timp pentru trecerea microprocesorului în regimul pas cu pas

Revenirea la programul MONITOR se poate face prin tasta *RESET*. Rularea normală a programului poate fi declanșată în orice moment prin tasta *RUN* care readuce bistabilul *BRUN* la 0.

În orice moment al rulării pas cu pas apăsarea tastei *INTROP* duce la înscrierea bistabilului *BINT* și aduce bistabilul *BRUN* la zero. Dacă întreruperea este validată se execută un ciclu de întrerupere prin care se citesc liniile de date care fiind pe 1 logic formează codul lui *RST7*. Prin aceasta se sare la subprogramul de salvare a registrelor microprocesorului în memoria *RAM*. Apoi cu ajutorul tastei *REG* pot fi vizualizate oricare din aceste registre. După efectuarea vizualizărilor se poate reveni la rularea programului utilizator din punctul în care a fost întrerupt prin comanda *REV*.

3.2.8. CONECTAREA TASTATURII TO-4

Semnalele pe care le furnizează tastatura *TO*-4 sînt codul tastei pe 5 bit, strobul tastaturii *CMIT* și cele 5 semnale de comandă *RESET*, *STEPI*, *STEPM*, *RUN*, *INTROP* al căror rol a fost descris în paragraful precedent.

Modul în care tastatura este conectată la schema microcalculatorului apare în schema bloc din figura 3.4. Pentru furnizarea de către tastatură a semnalelor amintite mai sus sînt necesare unele adaptări la nivelul circuitelor aparținînd electronicii proprii a tastaturii. Adaptarea nu modifică nimic în schema originală de fabricație.

În primul rînd se dotează tastatura cu un generator de tact deoarece prin construcție s-a prevăzut funcționarea acesteia cu un tact extern. Generatorul este realizat simplu în jurul unui circuit basculant bistabil Schmitt (fig. 3.12). Plăcuța pe care se află generatorul a fost introdusă în carcasa tastaturii.

Pe de altă parte impulsurile de comandă care acționează la nivelul schemei de stabilire a regimurilor sînt culese din schema tastaturii în punctele arătate în tabelul 3.4. În fine două semnale de intrare *LISTM* și *EXP* au fost legate la masă (0 logic) pentru a autoriza funcționarea schemei de explorare.

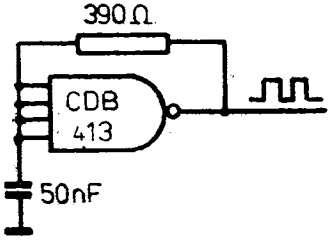


Fig. 3.12. Generator de impulsuri de tact pentru tastatura TO-H

Codul unei taste apăsate este furnizat pe cele cinci linii $RCT_1 - RCT_5$ (tab. 3.5). Anunțarea codului unei taste nou acționate se face de către semnalul de strob *CMIT*, cu excepția tastelor *RUN*, *STEPM* și *STEPI*. Ultimele două semnale logice sînt introduse într-o funcție SAU pentru a genera semnalul *STEP*.

Tabelul 3.4.

Punctele din schema TO-4 în care se culeg semnalele

SEMNAL	CONECTOR EXTERIOR	CONECTOR INTERN	CAPSULA	PIN
INTROP	16	TL 07	K1	7
RESET	09	TL 10	D3	9
RUN	24	TL 03	K1	2
STEPM	15	TL 02	K1	4
STEPI	17	TL 01	K1	3
LISTM	18	TL 13	LA MASĂ	
EXP	04	TL 20	LA MASĂ	

Tabelul 3.5.

Codurile generate de comenzile tastaturii TO-4

NUMELE TASTEI	CODUL TASTEI RCT5	RCTH	RCT3	RCT2	RCT1	STROB CMIT
ADR	1	1	0	1	0	DA
MEM	1	0	1	1	0	DA
RUN	1	0	0	1	1	DA
STEPI	1	0	1	0	1	NU
STEPM	1	0	1	1	1	NU
REV	1	0	0	0	1	DA
EXEC	1	1	0	0	0	DA
REG	1	0	1	0	0	DA
INTROP	1	1	1	0	1	DA
INCR	1	0	0	0	0	DA
RESET	1	1	1	1	1	DA
CLEAR	1	0	0	1	0	DA

Rezultă că prin program trebuie sesizate cele 5 semnale de cod $RCT_1 - RCT_5$ strobul *CMIT*, semnalele *STEP* și *RUN*, în total 8 semnale. Citirea lor se realizează printr-un circuit 8212 conectat ca port de intrare și selectat prin CS_2 și *I/OR*. Adresa portului tastaturii este 8000_{16}. (fig. 3.13).

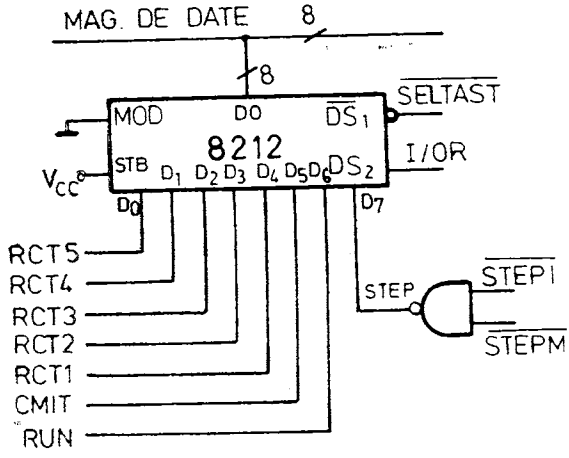


Fig. 3.13. Portul de conectare al tastaturii TO-4

3.2.9. PUNEREA LA PUNCT A MICROCALCULATORULUI

Aparatele destinate punerii la punct a unei scheme cu microprocesor au fiecare posibilităţile şi limitele lor. În general aparatele cu care se lucrează pentru evidenţierea unei funcţionări defectuoase sînt cele arătate în tabelul 3.6, în care se prezintă totodată şi problemele specifice ce pot fi analizate de fiecare. Primele trei aparate sînt mai obişnuite şi se află în mod curent în oricare laborator de electronică în timp ce ultimele sînt aparate pretenţioase care au apărut recent în dotarea laboratoarelor.

Tabelul 3.6.

Aparate destinate punerii la punct sau depanării schemelor cu microprocesoare în laborator

PROBLEME CE POT FI REZOLVATE:	AVO (MULTIMETRU NUMERIC)	SONDA LOGICĂ	OSCILO.	ANALIZOR SEMNĂTURI	ANALIZOR LOGIC	EMULATOR SOFTWARE (SIMULATOR)
scurtcircuite, întreruperi						
tensiuni greşite	da	eventual	da	nu	eventual	nu
rezistenţe, capacităţi defecte	da	nu	da	nu	nu	nu
nivele logice necunoscute	da	da	da	da	da	nu
pene software	nu	nu	eventual	nu	da	da

În cazul de față s-au folosit un multimetru numeric, un tranzistormetru, un testor de circuite integrate *TTL*, un osciloscop cu două spoturi avînd banda de frecvență de 50 MHz și un analizor logic.

În funcționarea unui sistem cu microprocesor există probleme care apar la punerea la punct cît și în timpul exploatării ce pot fi împărțite în patru categorii : greșeli de cablaj (trasee întrerupte, scurtcircuite), componente defecte sau cu valori diferite, pene de program și zgomote. Se va urmări modul de rezolvare a acestor probleme în cursul punerii la punct a microcalculatorului de dezvoltare.

Deși secvența de punere în funcțiune din fig. 3.14 prezintă trăsături particulare propii schemei pentru care a fost elaborată, ea păstrează elemente cu caracter general care pot fi puse în aplicare în situații asemănătoare.

În prima parte a secvenței de punere la punct s-au verificat riguros ambele fețe de circuit imprimat ale plăcii prin control vizual și comparație cu desenul cît și prin măsurarea cu ohmmetrul a rezistenței traseelor între pini ocazie cu care au fost depistate unele trasee lipsă și altele incomplete.

Apoi au fost controlate valorile rezistențelor și condensatoarelor, au fost sortate diodele luminoase și tranzistorii. Circuitele integrate TTL au fost verificate pe un testor de circuite integrate logice. După implantarea componentelor, cu excepția circuitelor LSI pentru care sînt prevăzute socluri, s-au efectuat noi controale vizuale ale lipiturilor și s-a trecut la partea a doua a punerii sub tensiune.

Nici unul din circuitele LSI (8080, 8224, 2102, 8212, 2708) *nu se află în soclu*, în prima fază a punerii sub tensiune.

Înainte de prima conectare a tensiunilor de alimentare ele trebuie verificate. Din cele trei tensiuni de alimentare ale lui 8080, ± 5 V și 12 V, aceea de -5 V nu absoarbe un curent mai mare de 1 mA și poate fi obținută cu o schemă simplă de redresare a tactului de 18 MHz (fig. 3.15).

Conform recomandărilor firmei producătoare tensiunea de 12 V *nu se va conecta niciodată înaintea* tensiunii de 5 V, iar la deconectare se va întrerupe mai întîi tensiunea cea mai ridicată (sau deodată cu cea de 5 V) [4].

În cadrul testelor sub tensiune semnalele furnizate de tastatură precum și altele care provin din sistem au fost generate de un grup de comutatoare (*HOLD, RESET, READY, INTROP*).

Cîteva momente mai semnificative din timpul punerii la punct vor fi mai pe larg prezentate. Astfel imediat după introducerea microprocesorului și conectarea alimentării, în condițiile în care $HOLD=1$, $READ=1$ și $RESET=0$ microprocesorul se va afla în starea inițială. Imediat după ce *RESET* este pus pe 1 logic microprocesorul intră în starea T_1 apoi T_2 moment în care se testează *HOLD* și *READY*. Mai întîi se testează *READ* și acesta fiind 1 logic se ajunge la testarea lui *HOLD* și deoarece acesta e și el logic are loc înscrierea fanionului intern destinat stării *HOLD*. Răspunsul *HLDA* care

VERIFICAREA TENSIUNILOR DE ALIMENTARE
DA SE CONECTEAZĂ TENSIUNEA +5 V
REMEDIERE ←—SCURTCIRCUITE, TENSIUNI APĂRUTE LA PINI NEPOTRIVIȚI ?
NU
SE REPETĂ OPERAȚIILE PENTRU TENSIUNILE +12 V ȘI −5 V
SE INTRODUCE ÎN SOCLU CAPSULA 8224
SE CONECTEAZĂ COMUTATOARELE LOGICE PENTRU RESET ȘI READYIN
SE CONECTEAZĂ TENSIUNILE +5 ȘI +12 V (RESET=0)
NU SE VIZUALIZEAZĂ SEMNALELE Φ 1, Φ 2 (MOS) ȘI Φ 2 (TTL)
REMEDIERE ←—PARAMETRI CORECȚI ?
DA
NU SE COMUTĂ READYIN ȘI RESET
REMEDIERE ←—RĂSPUNSURILE LUI 8224 SÎNT CORECTE ?
DA
SE DECONECTEAZĂ TENSIUNILE
SE INTRODUCE ÎN SOCLU 8080
SEMNALUL HOLD=1, READYIN=1, RESET=0
DA SE CONECTEAZĂ TENSIUNILE
REMEDIERE ←—SEMNALE SYNC ?
NU
NU SE PUNE RESET=1
REMEDIERE ←—SEMNALUL HLDA=1 ?
DA
NU SE PUNE HOLD=1 ȘI READYIN=0
REMEDIERE ←—SEMNALUL WAIT=1 ?
DA
NU SE PUNE READY=1
REMEDIERE ←—SEMNALUL SYNC APARE ? DAR DBIN, STSTB ?
DA
MICROPROCESORUL EXECUTĂ NEÎNTRERUPT RST7
SE INTRODUCE 8212 AMPLIFICATOR MAG. DATE INTRARE
ȘI SE FORȚEAZĂ LINIILE DE DATE LA MASĂ
MICROPROCESORUL EXECUTĂ NEÎNTRERUPT NOP-URI
NU SE PUNE RESET=0 PT. SCURT TIMP
REMEDIERI ←—LEDURILE AFIȘAJULUI DE ADRESĂ NUMĂRĂ ?
DA
NU
REMEDIERI ←—SÎNT CORECTE SEMNALELE CERAM, CS1, CS2 ?
DA
SE INTRODUC CIRCUITELE RAM
SE CONECTEAZĂ TASTATURA 0—4
SE INTRODUC RESTUL CIRCUITELOR 8212
NU SE INTRODUCE EPROMUL „MONITOR"
REMEDIERI ←—COMENZILE SE EXECUTĂ CORECT ?
DA
MICROCALCULATOR FUNCȚIONAL

Fig. 3.14. Secvența de punere în funcțiune a microcalculatorului

poate fi testat în exterior demonstrează că microprocesorul răspunde corect la cererile de acces direct la memorie.

Urmează apoi testarea în condițiile în care *HOLD*=0 iar *READY* este adus la 0 logic. Microprocesorul va ieși din starea *HOLD* și intră în starea T_W care se verifică prin punerea pe 1 logic a semnalului disponibil la pinul capsulei *WAIT*.

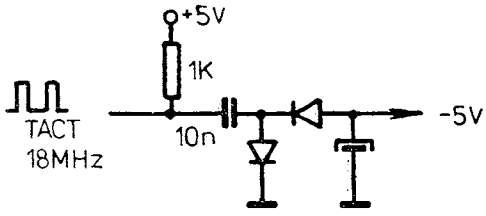


Fig. 3.15. Schema pentru obținerea tensiunii de alimentare —5 V a microprocesorului

Prin comutarea lui *READY* pe 1 logic are loc declanșarea rulării neîntrerupte a lui 8080. Funcționarea în acest regim se urmărește prin vizualizarea pe osciloscop a semnalelor *SYNC*, *STSTB*, *DBIN*. Deoarece cele două amplificatoare ale semnalelor de pe magistrala de date nu sînt introduse (8212), liniile de date ale microprocesorului sînt neconectate. De aceea în cadrul unui ciclu de instrucțiune se va citi întodeauna codul FF_{16} corespunzînd instrucțiunii RST7. Instrucțiunea fiind un *CALL* scurt, adică un apel de subprogram și adresa fiind una fixă 0038_{16}, efectul executării acestei instrucții va fi depunerea în stivă a conținutului registrului PC și saltul la adresa 0038_{16}. Dar și de data aceasta se va citi aceeași instrucție FF_{16} și în concluzie microprocesorul execută o buclă în jurul aceleiași instrucții.

O altă situație interesantă apare după introducerea în soclu a circuitului 8212 destinat intrării datelor de pe magistrală și forțarea la 0 logic a magistralei de date prin legare la masă. În noua împrejurare 8080 va citi mereu 00_{16} de pe liniile de date adică codul instrucției *NOP*. Efectul va fi nul doar că registrul PC își va mări conținutul cu unu după fiecare *NOP* executat. Întrucît amplificatorul pentru ieșirea datelor nu se află în soclu, conținutul registrului de stare va fi 0 logic peste tot, dar mai important e că $\overline{STACK}$=1. Prin urmare este autorizată afișarea dacă se forțează $\overline{WR}$ la zero logic. Acesta este motivul pentru care pe ledurile de adresă se poate urmări cum registrul PC, numără.

În fine cel mai important moment îl constituie conectarea tastaturii și apoi a memoriei *EPROM* în care se află MONITORUL. Acesta din urmă a fost elaborat în limbaj de asamblare INTEL 8080 și a fost rulat pentru un sistem FELIX-256 dispunînd de un cross-asamblor apoi parțial testat pe microcalculatorul prezentat în cap. 2.

În cele din urmă programul MONITOR a fost înscris într-o memorie EPROM 2708. Testarea finală a MONITORULUI s-a făcut în condițiile în care trebuie să funcționeze adică verificînd fiecare din comenzile de la tastatură, *ADR*, *MEM*, *STEPM* etc. Deoarece aceste comenzi au fost proiectate să facă uz de afișări, modul lor de desfășurare poate fi ușor controlat. Dacă toate

comenzile se execută corect nu mai devine necesară corecția programului din EPROM ci se poate trece la testări complexe luînd în considerare secvențe tipice de comandă ale unui utilizator pentru a observa comportarea de ansamblu.

În privința zgomotelor acestea se regăsesc pretutindeni dar nu este admisă depășirea unui nivel dincolo de care se perturbă grav funcționarea circuitelor logice. Au fost verificate impulsurile pe magistralele de date, adresă și de comandă precum și liniile de alimentare. Pe acestea din urmă pot apărea zgomote importante provocate de comutația simultană a mai multor circuite, ceea ce provoacă schimbări ale valorii tensiunii de alimentare.

3.2.10. MODUL DE OPERARE

Pentru a demonstra operarea microcalculatorului de către un utilizator se va considera un exemplu de program scurt care va fi încărcat și executat.

Programul (fig. 3.16) începe cu introducerea unei constante în acumulator apoi se mărește conținutul acestuia cu unu în mod repetat pînă cînd se trece de la valoarea FF_{16} la 00_{16} moment în care se execută oprirea. Instrucțiile în mnemonice ale limbajului de asamblare sînt arătate în tab. 3.7. Același program scris în limbaj mașină are înfățișarea din tab 3.8. Programul ocupă 7 octeți din memoria RAM de la adresa hexazecimală 2000_{16}. Testarea conținutului zero al acumulatorului se face de către instrucțiunea *JNZ*, salt condiționat de bistabilul *Z*.

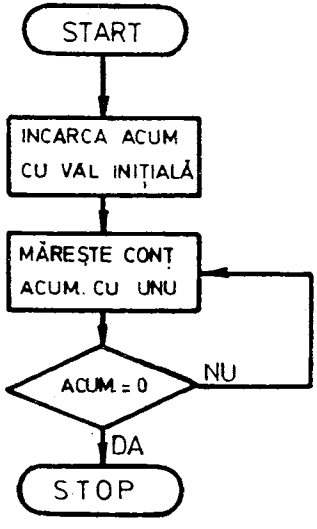


Fig. 3.16. Organigrama exemplului de program pentru demonstrarea operării.

În vederea executării programul trebuie încărcat mai întîi în memoria RAM. Se înțelege că acest lucru nu poate avea loc decît prin intermediul tastaturii.

INIȚIALIZAREA – RESET

Prin acționarea tastei RESET se întrerupe orice activitate a microprocesorului și numărătorul de adrese-*PC* este adus la zero. Eliberarea tastei declanșează lansarea MONITORULUI. La începutul programului MONITOR se inițializează registrul de 16 bit-*SP*, indicatorul stivei. Zona de memorie atribuită stivei începe la adresa 23 FF_{16}, adică la capătul memoriei RAM.

Tabelul 3.7.

MNEMONIC DE ASAMBLOR	COD HEXA	COMENTARII
MVI A, dd	3E X X	„move immediate t, A data XX" încărcarea directă a acumulatorului cu valoarea imediat următoare codului 3E
INR A	3C	„Increment A" A←A+1
JNZ Adr	C2 hhll	„jump on no zero to adress hhll", salt la adresa „hhll" dacă Z=0, altminteri treci la instrucţia următoare
HALT	76	instrucţia de oprire
	dd XX hhll	prescurtare care semnifică un cuvînt de date de 8 bit valoarea iniţială pe 8 bit cu care se încarcă acumulatorul adresă pe 16 bit formată din octetul superior hh şi cel inferior ll

Tabelul 3.8

ADRESA RAM	CONŢINUTUL RAM(HEX)	ETICHETA	COD ASAMBLOR		COMENTARII
2000	3E	START	MVI A, FD	;	SE ÎNCARCĂ ACUM CU VALOAREA FD_{16}
2001	FD			;	
2002	3C	CICLU	INR A	;	MĂREŞTE CU UNU ACUM
2003	C2		JNZ CICLU	;	SALT LA ETICHETĂ
2004	02			;	„CICLU" DACA
2005	20			;	DIST. Z=0. ALTMINTERI
				;	TRECE LA INSTRUCŢIA URMĂTOARE
2006	76		HALT	;	OPREŞTE PROGRAMUL

ÎNCĂRCAREA ADRESEI ŞI CITIREA UNEI LOCAŢII DE MEMORIE ADR

După apăsarea tastei *ADR*, programul MONITOR acceptă patru cifre hexazecimale reprezentînd adresa RAM (sau ROM) dorită de utilizator. După ultima cifră MONITORUL asigură afişarea adresei şi datelor din RAM. Totodată se permite lansarea unui program de la o adresă astfel încărcată. Formatul comenzii este *ADR* $XXXX_{16}$.

MODIFICAREA CONŢINUTULUI MEMORIEI – MEM

Conţinutul unei locaţii de memorie RAM a cărei adresă a fost afişată prin comanda *ADR* poate fi schimbat prin apăsarea tastei *MEM* urmată de cifrele

hexazecimale (două) reprezentînd octetul de date. Formatul comenzii este deci $MEMXX_{16}$.

INCREMENTAREA ADRESEI – INCR

Cu ajutorul tastei *INCR* se măreşte cu unu valoarea adresei introdusă prin *ADR* la fiecare apăsare şi concomitent se afişează şi octetul de date corespunzător din memorie. Se permite astfel vizualizarea memoriilor RAM şi ROM octet cu octet.

ÎNCĂRCAREA PROGRAMULUI DAT CA EXEMPLU

Secvenţa de acţionare a tastelor din tabelul 3.9 este destinată încărcării programului din fig. 3.16. Sînt arătate conţinutul afişajului de adresă şi date în fiecare moment. Se apasă mai întîi RESET. Se încarcă apoi adresa primei locaţii din RAM 2000_{16}, acţionînd tastele *ADR*, 2, 0, 0, 0. Apăsînd pe tasta *MEM* şi apoi pe 3 şi pe *E* se observă apariţia pe ledurile de date şi adresă a conţinuturilor $3E_{16}$ şi 2000_{16}. Asemănător se încarcă ceilalţi octeţi ai programului. După încărcarea ultimului octet se poate trece la vizualizarea încărcării corecte prin formarea adresei *ADR* 2000 şi apăsarea repetată a lui *INCR* (tab. 3.10).

Tabelul 3.9

Secvenţa de operare pentru încărcarea programului din tab. 3.8.

Nr. CRT.	TASTE ACŢIONATE	AFIŞAJ ADRESĂ	AFIŞAJ DATE
1	RESET	– – – –	– – – –
2	ADR 2 0 0 0	2 0 0 0	X X
3	MEM 3 E	2 0 0 0	3 E
4	MEM F D	2 0 0 1	F D
5	MEM 3 C	2 0 0 2	3 C
6	MEM C 2	2 0 0 3	C 2
7	MEM 0 2	2 0 0 4	0 2
8	MEM 2 0	2 0 0 5	2 0
9	MEM 7 6	2 0 0 6	7 6

Tabelul 3.10.

Secvenţa de operare pentru vizualizarea zonei de memorie în care se află programul

NR. CRT.	TASTE ACŢIONATE	AFIŞAJ ADRESĂ	AFIŞAJ DATE
1	RESET	– – – –	– – – –
2	ADR 2 0 0 0	2 0 0 0	3 E
3	INCR	2 0 0 1	F D
4	INCR	2 0 0 2	3 C
5	INCR	2 0 0 2	C 2
6	INCR	2 0 0 4	0 2
7	INCR	2 0 0 5	2 0
8	INCR	2 0 0 6	7 6

VIZUALIZAREA REGISTRELOR – REG

Posibilitatea vizualizării tuturor registrelor de date ale unității centrale de microprocesor constituie un element major de mare utilitate în punerea la punct a unui program.

Vizualizarea conținuturilor registrelor se poate face în timpul rulării normale sau în regimurile de funcționare pas cu pas. Înainte de a vizualiza un registru se dă întotdeauna o întrerupere de operator prin tasta *INTROP*, Alegerea registrului dorit se obține prin tastele *A*, *B*, *C*, *D*, *E*, F, 1 (H), 2 (*L*), 4 (*PCH*), 5 (*PCL*).

Mecanismul prin care se recunoaște întreruperea de operator este următorul : apăsarea tastei *INTROP* provoacă înscrierea fanionului *BINTR* iar *BRUN* este forțat pe zero deci pentru regimul de rulare normală. Recunoașterea întreruperii de către microprocesor care are validată întreruperea de către o instrucțiune EI executată de MONITOR, este urmată de generarea semnalului $\overline{INTA}$ în cadrul octetului de stare. Acesta va șterge întreruperea memorată în *BINT* și în continuare subprogramul de deservire a întreruperii va salva conținutul registrelor în memorie de unde pot fi vizualizate. Revenirea la programul utilizator și în regimul care a precedat întreruperea se obține prin tasta *REV*.

REGIMUL DE RULARE NORMALĂ – RUN

Pentru rularea normală a unui program utilizator operatorul trebuie să apese tasta *RUN*. În acest regim nu se mai pot urmări afișajele de adresă și date. Microcalculatorul execută programul utilizator iar programul MONITOR este total inactiv. Revenirea lui MONITOR se realizează prin tasta *RESET*

REGIMUL DE RULARE PAS CU PAS – STEPI, STEPM

După acționarea oricăreia din tastele *STEPM* sau *STEPI*, 8080 va lansa programul utilizator în regim de rulare pas cu pas. Fiecare pas poate fi declanșat de una din tastele *STEPI* sau *STEPM*. În funcție de tasta apăsată se execută o instrucție sau un singur ciclu de mașină. Cele două taste pot fi acționate în orice ordine și în orice moment al executării programului. Acest lucru poate fi observat în tab. 3.11, în care declanșarea regimului de rulare pas cu pas s-a făcut cu *STEPM*. După această acțiune afișajul arată 6000_{16} (adresa 2000_{16} cu bitul A_{14}=1). Pe diodele luminiscente aparținînd afișajului de date apare 3*E* corespunzător instrucțiunii *MVI*. Dacă se dorește executarea acestei instrucțiuni care necesită două cicluri de mașină se poate apăsa *STEPI*. Dacă dorim doar aducerea codului instrucției 3E apăsăm *STEPM*. Adresa afișată va fi 6001 iar pe date *FD*. După apăsarea din nou a lui *STEPM* se aduce și acest octet și execuția e terminată deoarece ledul registrului de stare corespunzător lui M_1 este aprins semn că s-a început o nouă instrucțiune. Dealtfel pe ledurile

Tabelul 3.11.

Executarea programului utilizator în regim pas cu pas

NR. CRT.	TASTE ACȚIONATE	AFIȘAJ ADRESĂ	AFIȘAJ DATE
1	RESET	— — — —	— — —
2	ADR 2 0 0 0	2 0 0 0	3 E
3	STEPM	6 0 0 0	3 E
4	STEPM	6 0 0 1	F D
5	STEPM	6 0 0 2	3 C
6	STEPM	6 0 0 3	C 2
7	STEPM	6 0 0 4	0 2
8	S T E P M	6 0 0 5	2 0
9	STEPM	6 0 0 2	3 C
10	STEPI	6 0 0 3	C 2
11	STEPI	6 0 0 2	3 C
12	STEPI	6 0 0 3	C 2
13.	INTROP	—	—
14	REG A	—	F F
15	REV	6 0 0 2	3 C
16	STEPI	6 0 0 3	C 2
17	STEPI	6 0 0 6	7 6
18	STEPI	F F F F	F F *

* După instrucția HALT ambele afișaje au becurile aprinse

de date apare codul 3C corespunzător instrucției *INC A*, iar afișajul de adresă indică 6002_{16}.

Acum se apasă *STEPM* sau *STEPI* pentru că această instrucțiune *INC A*, are un singur octet. Se ajunge astfel la instrucțiunea *JNZ*, codul *C2*, adresa 6003_{16}. Prin apăsarea tastei *STEPI* se declanșează toate cele trei cicluri necesare acestei instrucțiuni și pentru că în acumulator se găsește *FE* se execută saltul înapoi la instrucțiunea *INC*, adică la adresa 6002 și codul 3C. Printr-o nouă apăsare a lui *STEPI* se ajunge iar la *JNZ* și tot prin *STEPI* din nou la *INC* de data aceasta acumulatorul avînd FF_{16}. După o nouă incrementare a lui A se aduce bistabilul de condiție Z pe 1 logic. În această situație se ajunge la codul instrucțiunii *HALT*, după care tastele *STEPM* sau *STEPI* nu mai au efect.

3.2.11. PROGRAMUL MONITOR

După cum s-a arătat, programul MONITOR aflat în memoria fixă EPROM este destinat realizării funcțiilor cerute de utilizator de la tastatură.

Fiecare comandă dată se execută în cadrul unei secțiuni a programului MONITOR organizată sub forma unui subprogram purtînd numele tastei acționate de exemplu *ADR*, *MEM*, *INCR*, ș.a.

Pentru a se ajunge la subprogramul corespunzător unei anumite taste partea de program principal a MONITOR-ului permite microcalculatorului să preia codul tastei furnizat de tastatură și să determine prin decodificare adresa subprogramului la care se va executa saltul. Odată cu executarea acțiunilor cerute de comandă, microcalculatorul se reîntorce la programul de decodificare în vederea introducerii, decodificării și executării altei comenzi [6].

Instrucțiunile pentru programul decodificator sînt listate în tab. 3.12. La adresa *DECOM*, se apelează subprogramul CITTAST pentru introducerea codului trimis de tastatură. Revenirea din subprogramul CITTAST se face numai în cazul acționării unei taste adică apariția semnalului de strob *CMIT*. Cînd microprocesorul revine în programul principal caracterul pe 5 bit transmis de tastatură se află în registrul A.

Tabelul 3.12.

Decodificarea comenzilor în programul MONITOR

```
;  SECVENȚA DIN PROGRAMUL MONITOR CARE ASIGURĂ DECODIFICAREA
;    COMENZILOR DE LA TASTATURĂ
   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                .
                .
                .
DECOM: CALL     CITAS     ; SE APELEAZĂ SUBPROGRAMUL DE CI
                          ; TIRE A PORTULUI TASTATURII. LA REVE-
                          ; NIRE CODUL
       STC                ; TASTEI ACȚIONATE SE AFLĂ ÎN ACU-
       CMC                ; MULATOR.CARRY=0
       RAR                ; SE DEPLASEAZĂ BITUL RCT5 ÎN CARRY.
       JNC      DECOM     ; DACĂ RCT5=0 NU ACCEPTĂ
       CPI      0AH       ; SE COMPARĂ CU CODUL TASTEI ADR
       JZ       ADR       ; SALT LA SUBPROGRAMUL DE REALIZARE
                          ; A FUNCȚIEI ADR
       CPI      06H       ; SE COMPARĂ CU CODUL TASTEI MEM
       JZ       MEM       ; SALT LA SUBPROGRAMUL FUNCȚIEI
                          ; MEM
       CPI      00H       ; SE COMPARĂ CU CODUL TASTEI INCR
       JZ       INCR      ; SALT LA SUBPROGRAMUL FUNCȚIEI INCR
       CPI      04H       ; SE COMPARĂ CU CODUL TASTEI REG
       JZ       REG       ; SALT LA SUBPROGRAMUL FUNCȚIEI REG
       CPI      05H       ; SE COMPARĂ CU CODUL TASTEI STEPI
       JZ       STEPI     ; SALT LA SUBPROGRAMUL FUNCȚIEI STEP
       CPI      03H       ; SE COMPARĂ CU CODUL TASTEI RUN
       JZ       RUN       ; SALT LA SUBPROGRAMUL FUNCȚIEI RUN
       CPI      01H       ; SE COMPARĂ CU CODUL TASTEI REV
       JZ       REV       ; SALT LA SUBPROGRAMUL FUNȚIEI REU
       JMP      DECOM     ; SE REVINE LA INSPECTAREA TASTATURII
```

Tabelul 3.13.

Subprogramul de realizare a funcției ADR

```
;  SUBPROGRAMUL PERMITE INTRODUCEREA ADRESEI PRIN PATRU CIFRE
;  HEXAZECIMALE ŞI AFIŞAREA LOCAŢIEI DE MEMORIE ADRESATE
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
ADR :   CALL    CITAS     ;  SE APELEAZĂ SUBPROGRAMUL DE CITIRE
                          ;  A PORTULUI TASTATURII
        RRC               ;  SE DEPLASEAZĂ RCT5 ÎN CARRY. DACA
                          ;  RCT5=
        JNZ     ADR       ;  1 SE REIA CITIREA TASTATURII
        ANI     OFH       ;  SE APLICĂ MASCA PENTRU PRIMII 4 BIŢI
                          ;  DIN ACUMULATOR
        RLC               ;  PATRU
        RLC               ;  DEPLASĂRI
        RLC               ;  LA STÎNGA
        RLC               ;  ÎN ACUMULATOR
        MOV     H, A      ;  TRANSFER ÎN REGISTRUL H
HEX 2:  CALL    CITAS     ;  SE APELEAZĂ SUBPROGRAMUL DE CITIRE
                          ;  A PORTULUI TASTATURII
        RRC               ;  SE TESTEAZĂ RCT5 IN
        JNZ     HEX 2     ;  CARRY
        ANI     OGH       ;  SE APLICĂ MASCA PENTRU PRIMII 4 BIŢI
                             DIN ACUMULATOR
        ORA     H         ;  SE FORMEAZĂ OCTETUL COMPLET PRIN
                          ;  SAU LOGIC
        MOV     H, A      ;  SE DEPUNE OCTETUL ÎN H
HEX3 :  CALL    CITAS     ;          ↑
        RRC               ;          |
        JNZ     HEX 3     ;          |
        ANI     OGH       ;          |
        RLC               ;          |
        RLC               ;          |
        RLC               ;  CA MAI SUS DAR SE ASAMBLEAZĂ OCTETUL
        RLC               ;  ÎN REGISTRUL L
        MOV     L, A      ;          |
HEX 4:  CALL    CITAS     ;          |
        RRC               ;          |
        JNZ     HEX 4     ;          |
        ANI     OFH       ;          |
        ORA     L         ;          |
        MOV     L, A      ;          ↓
        MOV     A, M      ;  SE CITEŞTE LOCAŢIA ADRESATĂ
        MOV     M, A      ;  SE DEPUNE CONŢINUTUL ŞI ARE
        JMP     DECOM     ;  LOC VIZUALIZAREA DATELOR
```

Se testează mai întîi valoarea bitului *RCT*5 aflat în rangul c.m.p.s. al registrului *A*. Dacă acesta are valoarea 1 logic înseamnă că s-a acționat o tastă de comandă, în caz contrar este vorba de o tastă pentru date care va fi ignorată.

Dacă registrul *A* are memorată o comandă conținutul acestuia se compară succesiv cu valorile binare ale codurilor tastelor *ADR*, *MEM*, *INCR* etc.

Dacă codul tastei apăsate este egal cu oricare din aceste valori, microprocesorul execută instrucția de salt la subprogramul ce corespunde comenzii cerute.

În continuare se exemplifică modul în care se execută o comandă. Presupunem că s-a acționat tasta „*ADR*". Deoarece după apăsarea tastei *ADR* operatorul trebuie să transmită adresa de memorie dorită, exprimată sub forma a 4 cifre hexazecimale, subprogramul „*ADR*" va citi aceste cifre și le va asambla în registrele *H* și *L*.

Examinînd subprogramul *ADR*, listat în tab. 3.13 se observă cele 4 apeluri ale subprogramului *CITAST*. După prima apelare, tetrada primei cifre hexazecimale recepționate este deplasată în poziția $A_7—A_4$ a acumulatorului iar biții $A_3—A_0$ sînt puși pe zero, și apoi rezultatul e memorat în registrul *H*.

Următoarei tetrade recepționate i se aplică de asemeni o mască pentru a aduce la zero biții $A_7—A_4$ și printr-o operație logică *SAU* cu registrul *H* se obține octetul superior de adresă care va fi plasat în registrul *H*.

Asemănător se procedează pentru asamblarea octetului inferior al adresei în registrul *L* folosind următoarele două tetrade transmise de la tastatură. Pentru că ADR $XXXX_{16}$ este o comandă care permite și vizualizarea locației de memorie adresate, subprogramul citește conținutul locației și apoi îl reînscrie în cursul a două instrucțiuni *MOV A, M* și *MOV M, A*. După aceasta din urmă conținutul binar al locației adresate va fi vizualizat de către registrul de date care comandă diodele luminiscente de date. În fine ultima acțiune este revenirea la adresa *DECOM*, microprocesorul fiind astfel pregătit pentru o nouă comandă.

3.3. INTERFAȚA CU LECTORUL DE BANDĂ LB-50

3.3.1. PREZENTAREA LECTORULUI

Banda perforată este un suport de informații care are o largă răspîndire fiind prezentă în continuare în echipamentele moderne de calcul, de transmiterea datelor și în cele de automatizare. Față de cartela perforată, principalul suport concurent, de partea benzii există un avantaj mare acela că echipamentele cu bandă sînt mai ieftine. Or se știe că, un criteriu esențial de

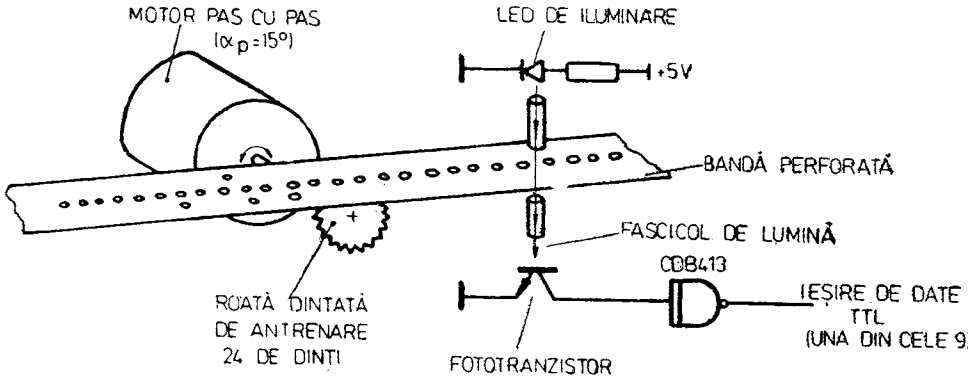


Fig. 3.17. Principiul de funcționare al lectorului de bandă LB-50

alegere a echipamentelor periferice destinate microcalculatoarelor este prețul de cost care trebuie să fie cît mai scăzut. Iată de ce lectorul și perforatorul de bandă sînt periferice preferate în sistemele cu microprocesor.

Un echipament destinat citirii benzilor perforate cu 8 piste de informații este lectorul *LB*-50 (IEPER—București). [7]. Antrenarea benzii la acest lector se face de către o roată dințată cuplată la un motor electric pas cu pas (fig. 3.17). În permanență unul din cei 24 de dinți ai roții pătrunde în cîte o perforație a pistei de sincronizare. Sesizarea perforațiilor benzii se realizează fotoelectric prin iluminarea benzii de către un grup de 9 diode luminoase. În dreptul diodelor există nouă fototranzistori care în momentul trecerii perforațiilor detectează prezența luminii. Singurul fototranzistor iluminat indiferent de caracter aparține pistei de sincronizare care are în mod obligatoriu perforația de sincronizare. După fiecare pas al motorului, banda se deplasează cu un caracter. Cu această ocazie are loc o succesiune lumină-întuneric-lumină care permite fototranzistorului să emită un impuls electric ce va fi format și care va declanșa un monostabil. Impulsul de la ieșirea monostabilului se numește *SINCRO*.

El apare mai tîrziu decît eventualele impulsuri de date furnizate de pistele de date deoarece diametrul perforației de antrenare este de 1,2 mm în timp ce pentru date este 1,6 mm. Acest lucru este de natură să permită folosirea impulsului SINCRO ca semnal de strobare a liniilor de date.

3.3.2. SCHEMA INTERFEȚEI

Pentru conectarea lectorului la microcalculator este necesară proiectarea unei interfețe care să îndeplinească condițiile de compatibilitate la nivelul vitezei, diagramei de timp a semnalelor lectorului (fig. 3.18) și microcalcula-

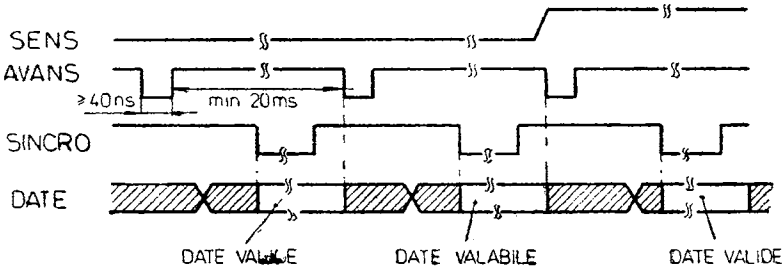


Fig. 3.18. Diagrama de timp a semnalelor lectorului LB-50

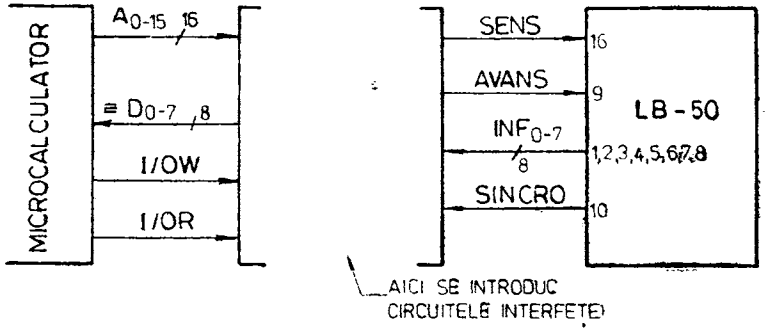


Fig. 3.19. Semnalele interfeței lectorului LB-50 cu microcalculatorul

torului, structurii datelor, caracteristicilor electrice ale semnalelor. Acestea din urmă sînt analizate în cadrul tabelului 3.14. Se observă că nu apar probleme de adaptare toate semnalele avînd nivele *TTL*. Interfața primește și emite semnalele arătate în figura 3.19. Realizarea interfeței se reduce deci la rezolvarea problemelor funcționale. În acest scop analiza funcțională indică următoarele funcții : 1) asigurarea ritmului de lectură de 50 car/s, 2) încărcarea octetului de date în acumulator, 3) asigurarea sensului de citire și regimului *START-STOP*.

Aspectele funcționale de mai sus sînt rezolvate în cadrul interfeței prin circuite Fig. 3.20 și pe de altă parte prin programul de gestiune al lectorului executat în microprocesor.

Semnalul *SENS*, care determină direcția de deplasare a benzii perforate, asigură derularea benzii în ambele direcții. Derularea înapoi (convențional) este utilă pentru reluarea citirii în caz de eroare la citire sau pentru lectura integrală a benzii. Nivelul logic al semnalului *SENS* este memorat în bistabilul BSENS conectat la linia D_0. Poziționarea lui BSENS se face printr-o instruc-

Tabelul 3.14.

Analiza funcţională pentru semnalele lectorului LB-50

NUME SEMNAL	NR. BIT	DURATA (F−1)	CARACTERIST. ADAPTARE ELECTRICE		FUNCŢIA SEMNALULUI
SENS	1	nivel	TTL	—	indică sensul de mişcare a benzii
AVANS	1	min 50 sus	TTL	—	pe tranziţia 1→0 bandă avansează cu un caracter
INF (0−7)	8	20 ms	TTL	—	biţii de date
SINCRO	1	1−12 ms	TTL	—	marchează sosirea unei perforaţii de sincronizare în staţia de lectură

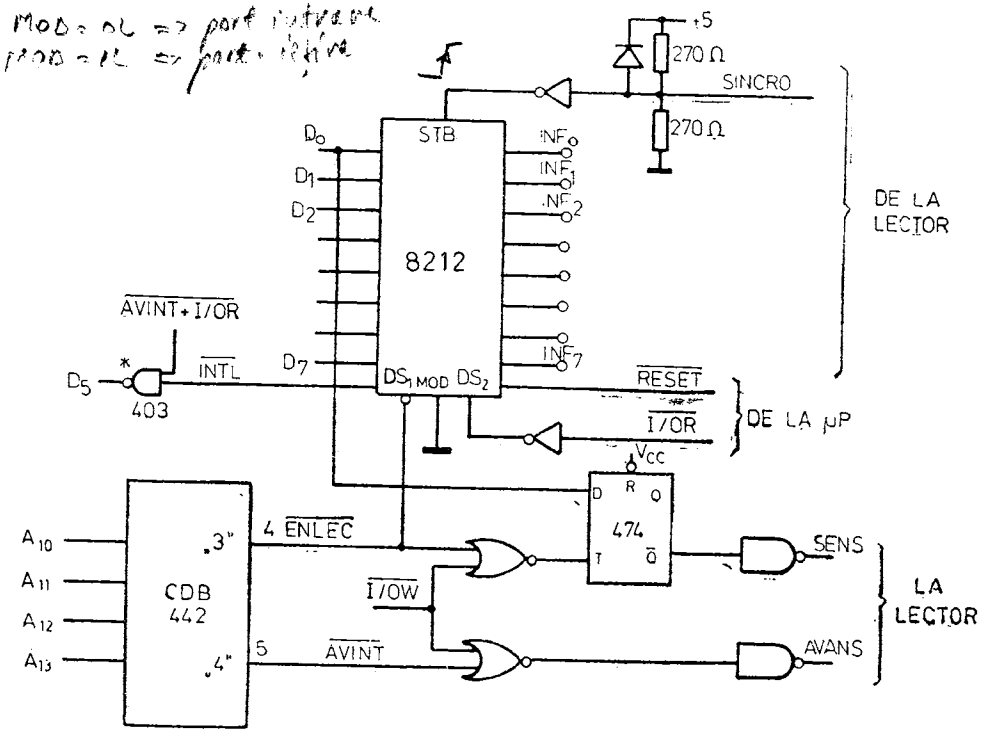


Fig. 3.20. Schema logică a interfeţei lectorului LB-50 cu microcalculatorul

ție de ieșire care să genereze $\overline{ENLEC}$ $\overline{I/OW}$ și prin D_0=1 sau 0. Pentru a genera impulsul $\overline{AVANS}$ se folosește ecuația logică $\overline{I/OW}+\overline{AVINT}$. După complementare el este trimis spre lector și pe tranziția 1 spre 0 declanșează mișcarea benzii. Durata impulsului este determinată de durata semnalului *I/OW*, de 500 ns. Semnalul de selecție $\overline{AVÎNT}$, se obține prin decodificarea biților $\bar{A}_{10}.\bar{A}_{11}.\bar{A}_{12}.\bar{A}_{13}$ la ieșirea decodificatorului *CDB* 442. În același circuit de decodificare se obține semnalul $ENLEC=A_{10}.A_{11}.\bar{A}_{12}.\bar{A}_{13}$ care se generează pentru ca împreună cu I/OW să poziționeze BSENS iar împreună cu *I/OR* să selecteze portul 8212 pentru citirea datelor. Circuitul 8212 este folosit ca port de intrare pentru memorarea datelor furnizate de lector ($INFO_{0-7}$. Memorarea se petrece odată cu tranziția 0 la 1 a semnalului *SINCRO* care acționează la intrarea de strobare *STB*. Dacă bistabilul fanion al circuitului 8212 este pe 1 logic, după venirea lui *SINCRO* el ajunge pe 0 logic și *INT* devine și el 0 logic. Starea bistabilului fanion SR poate fi adusă în acumulator printr-o instrucție de intrare care generează semnalul de selecție *AVÎNT* și *I/OR* prin care se deschide poarta cu colector în gol 403 și nivelul logic la lui $\overline{INTL}$ intră pe linia de date D_5. Prin testarea bitului A_5 al acumulatorului se poate decide dacă este sau nu cazul să se citească octetul din registrul 8212. Odată cu citirea se aduce fanionul *SR* pe 1 logic pregătindu-se astfel o nouă operație de citire la lector.

3.3.3. SUBPROGRAMUL DE CITIRE

Subprogramul de gestiune al lectorului pentru citirea unui număr de octeți și depunerea în memoria de date RAM de la o adresă anumită este arătat în tabelul 3.15 și în organigrama din fig. 3.21.

Subprogramul este conceput ca o subrutină căreia i se transmit cei doi parametri necesari, numărul de octeți de citit și adresa de depunere în RAM, de către programul principal. Pentru o înțelegere mai ușoară a programului se precizează că stabilirea sensului de deplasare a benzii se face prin instrucția *OUT* 8C, prin intermediul căreia bistabilul *BSENS* va fi încărcat cu valoarea bitului c.m.p.s. al acumulatorului. Selecția și introducerea octetului aflat în portul de intrare 8212 are loc prin instrucția *IN* 8C (același cod de selecție ca pentru *BSENS* dar instrucția de intrare). Fanionul *SR* este citit prin instrucția *IN* 90 și după aplicarea măștii 20_{16} se poate testa valoarea bitului 5 al acumulatorului. Impulsul de declanșare a mișcării benzii este *AVANS* și emiterea lui se efectuează prin instrucția *OUT* 90.

Tabelul 3.15

Subprogram de citire pentru LB-50

```
;  SUBPROGRAMUL POATE EFECTUA INTRODUCEREA UNUI NUMĂR VARIABIL
;  DE OCTEŢI (MAX. 256) DE PE BANDA PERFORATĂ ŞI DEPUNE ÎN MEMORIE
;  DE LA O ADRESĂ DATĂ
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

ADR1    EQU     20A0H   ;  ADRESA ZONEI DE MEMORIE UNDE SE
                        ;  INTRODUC OCTEŢI
ADNRO   EQU     20A2H   ;  LOCAŢIA DE MEMORIE UNDE SE AFLĂ
                        ;  NUMĂRUL DE OCTEŢI
FAN     EQU     90H     ;  ADRESA PENTRU GENERAREA IMPULSULUI
                        ;  AVANS SAU PENTRU SELECTAREA PORTU-
                        ;  LUI DE INTRARE FANIOANE
PORT    EQU     8CH     ;  ADRESA BISTABILULUI DE MEMORAREA
                           SENSULUI ŞI A PORTULUI DE INTRARE

;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

LECT:   MUI     A, 00H  ;  SE PREGĂTEŞTE SENSUL DE MIŞCARE AL
                           BENZII
        OUT     PORT    ;  SE STABILEŞTE SENSUL MIŞCĂRII
        LHLD    ADR1    ;  SE ÎNCARCĂ ÎN PERECHEA HL ADRESA
                        ;  ZONEI RAM
        LDA     ADNRO   :  SE ÎNCARCĂ NUMĂRUL DE OCTEŢI
        MOV     B, A    ;  SE TRANSFERĂ ÎN B
        CPI     00H     ;  NUMĂRUL DE OCTEŢI=0 ?
FINE:   RZ              ;  DACĂ DA SE REVINE
        OUT     FAN     ;  SE EMITE IMPULSUL AVANS
SIN:    IN      FAN     ;  SE CITEŞTE FANIONUL SR AL PORTULUI
                        ;  8212
        ANI     20H     ;  SE APLICĂ MASCA
        CPI     20H     ;  ESTE BITUL=1 ?
        JNZ     SINC    ;  DACĂ NU SE AŞTEAPTĂ CA SĂ DEVINĂ ZERO
        IN      PORT    ;  CITEŞTE OCTETUL
        XRI     OFFH    ;  SE COMPLEMENTEAZĂ DATELE
        MOV     M,A     ;  SE DEPUNE ÎN MEMORIA RAM
        INX     H       ;  MĂREŞTE CU UNU REG. HL
        DCR     B       ;  DECREMENTEAZĂ NR. OCTEŢI
        JMP     FINE    ;  SALT LA ETICHETA FINE
```

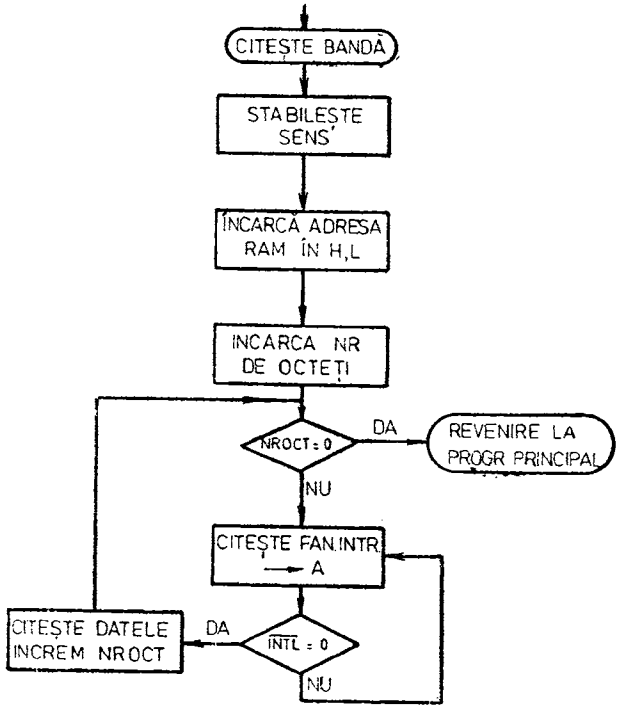


Fig. 3.21. Organigrama subprogramului de citire

3.4. INTERFAȚA CU PERFORATORUL DE BANDĂ P-50

3.4.1. PREZENTAREA PERFORATORULUI

Echipamentul *P*-50 (produs de IEPER-București) este destinat perforării benzilor cu 5, 6, 7, sau 8 piste de informație la viteza maximă de 50 de caractere pe secundă [8].

Echipamentul posedă două regimuri de lucru automat și manaual. În regim manual se pot comanda operații de deplasare a benzii înainte cu perforarea pistei de sincronizare, de deplasare înapoi, și deplasare înainte cu perforarea fiecărei piste. În regim automat perforatorul înregistrează pe bandă caracterele primite de la calculator în momentul primirii comenzilor corespunzătoare.

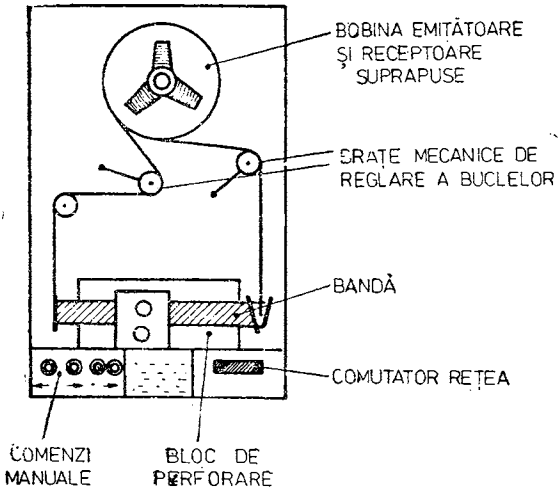


Fig. 3.22. Perforatorul de bandă P-50

Partea mecanică a perforatorului (fig. 3.22) cuprinde bobina emițătoare și bobina receptoare suprapuse și antrenate de același motor, sistemul de amortizare a șocurilor de antrenare a benzii și blocul de perforare. Acesta din urmă dispune de un motor propriu de antrenare și perforare. Principiul de realizare a perforării în acest bloc este ilustrat în figura 3.23. Pe axul motorului se află cama 1 care se poate roti în interiorul plăcii oscilante 2. Dacă se acționează electromagnetul 3, armătura 4 este atrasă și capătul 5 numit levier declanșator se deplasează între levierul de perforare 6 și ciocănelul 7. Perforarea se produce prin lovirea ansamblului amintit, de către placa oscilantă prin capătul acesteia *A*. După perforare electromagnetul își pierde alimentarea și arcul declanșatorului readuce piesa 5 în poziția inițială în timp ce rotirea în continuare a camei provoacă revenirea levierului de perforare sub acțiunea capătului B al plăcii oscilante.

Este evident că funcționarea mecanismului de perforare trebuie sincronizată cu partea electronică a echipamentului. Aceasta din urmă conține logica de comandă a perforării și schema pentru generarea semnalelor spre calculator

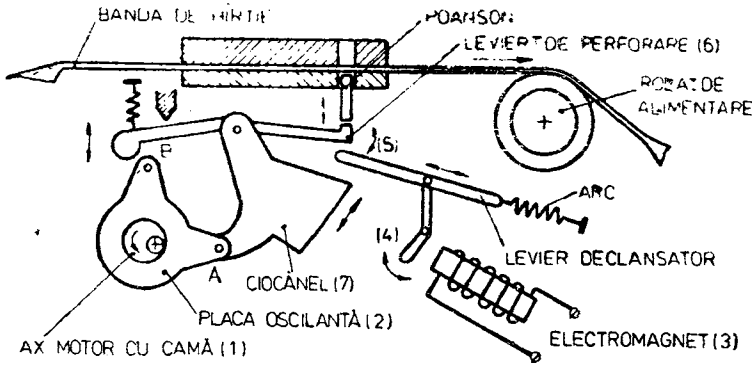


Fig. 3.23. Principiul de realizare a perforării

și primirea comenzilor. Perforatorul dispune de asemenea de surse de alimentare proprii.

Analiza funcțională arată că interfața trebuie să execute următoarele funcțiuni : 1) activarea perforatorului, 2) comanda perforării, 3) schimbarea logicii de perforare.

3.4.2. SCHEMA INTERFEȚEI

Diagrama de timp a semnalelor de comandă ale perforatorului este arătată în fig. 3.24. Pentru o funcționare corectă a perforatorului interfața (fig. 3.25) trebuie să asigure atît semnalele de comandă spre perforator cît și dialogul cu microprocesorul.

Analiza semnalelor perforatorului apare în tabelul 3.16. Perforatorul mai dispune și de alte semnale care pot fi incluse în interfață dacă se dorește acest lucru de către proiectant. În cazul de față semnalele alese asigură un minim de funcțiuni.

Comanda de perforare *CP*, declanșează ciclul de perforare pe tranziția 1→0 și trebuie menținută la nivelul logic 0 pînă în momentul cînd *RBPER*, confirmarea executării perforării devine 0 logic.

Perforarea se poate executa numai dacă perforatorul este activat. Comanda de activare *NPACT*=0 asigură pornirea motorului blocului de perforare. Dacă motorul a fost în repaus (*NPACT*=1) și se dă comanda de activare trebuie să se scurgă minimum 2 secunde pînă la atingerea turației nominale.

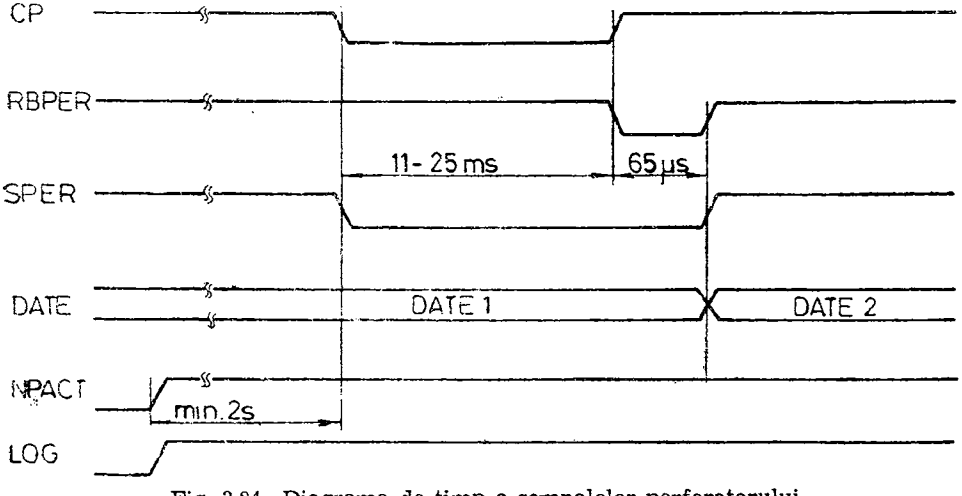


Fig. 3.24. Diagrama de timp a semnalelor perforatorului

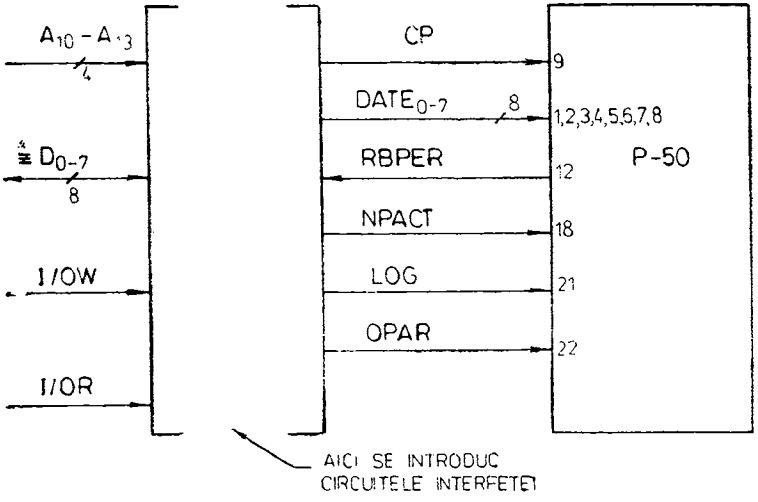


Fig. 3.25. Semnalele interfeței perforatorului P-50 cu microcalculatorul

Nivelul logic al semnalului *LOG* trebuie de asemenea determinat de microprocesor pentru a da posibilitatea programatorului să schimbe logica de perforare (1 să însemne perforație sau absență de perforație).

Întrucît se dorește perforarea pe 8 biți și nu într-un cod anumit (*ASCII, ISO, EBCDIC*) plus paritate, nivelul semnalului *OPAR* de inhibare a contro-

Tabelul 3.16.

Analiza semnalelor perforatorului P- 50

NUME SEMNAL	NR. BIT	DURATA (F_{-1})	CARACTERISTICI ELECTRICE	ADAPTARE	FUNCȚIA SEMNALULUI
D_{1-8}	8	50 Hz^{-1}	TTL	—	reprezintă cei 8 bit ai unui caracter de perforat
NPACT	1	NIVEL	TTL	—	cînd=0 activează motorul blocului de perforare
LOG	1	NIVEL	TTL		logica de perforare
OPAR	1	NIVEL	TTL	—	anularea parității cînd OPAR=1
CP	1	11—25 ms	TTL	—	comanda pentru perforare CP decl. perforarea 1→0
RBPER	1	65 sus	TTL	—	banda e perforată

lului de paritate trebuie și el decis de proiectant. Pentru inhibare el va fi 1 logic.

Schema logică a interfeței este dată în fig. 3.26. Se observă că nivelele semnalelor *NPACT* și *LOG* sînt stabilite de ieșirile registrului 495 care poate fi încărcat de pe magistrala de date prin semnalele $\overline{ADNPL}$ și *I/OW*. Conectarea celor 8 semnale de date *DATE* 1—*DATE* 8 la magistrala de date a microprocesorului se realizează cu un circuit 8212 funcționînd ca registru de ieșire. Bistabilul *SR* este folosit ca fanion. Răspunsul perforatorului RBPER este adus la intrarea *STB* și asigură înscrierea fanionului. Caracterul de perforat se face

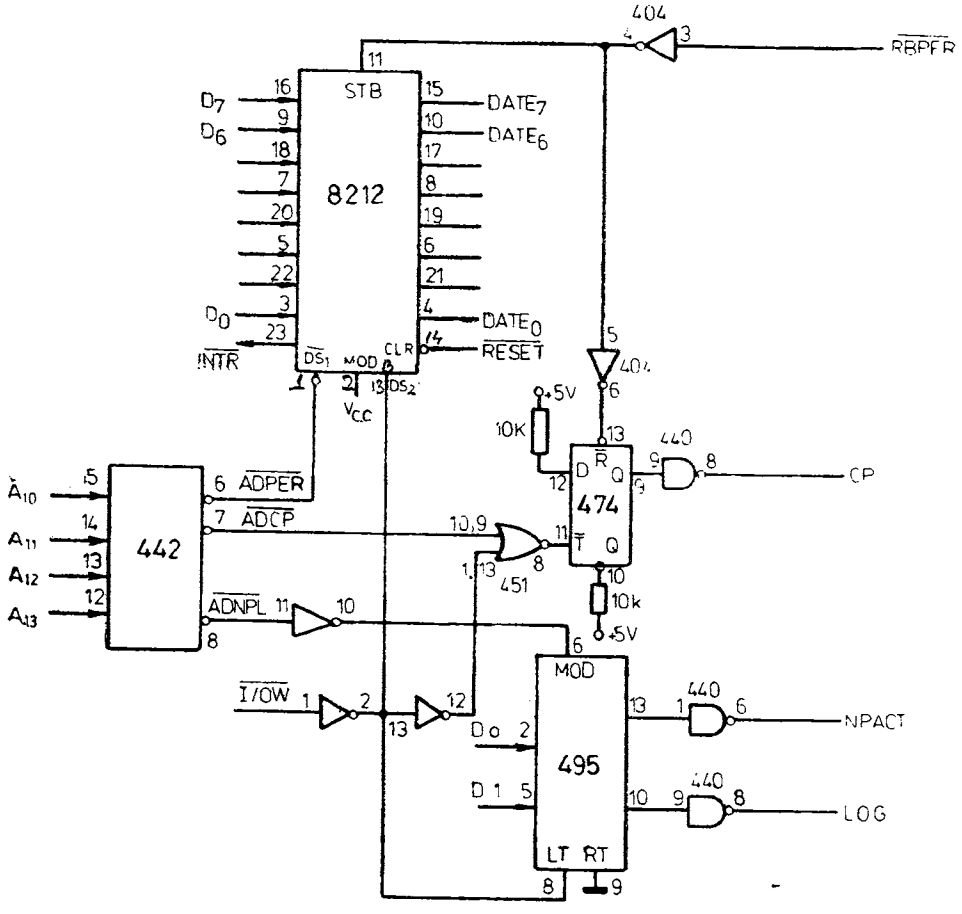


Fig. 3.26. Schema logică a interfeței perforatorului P-50 cu microcalculatorul

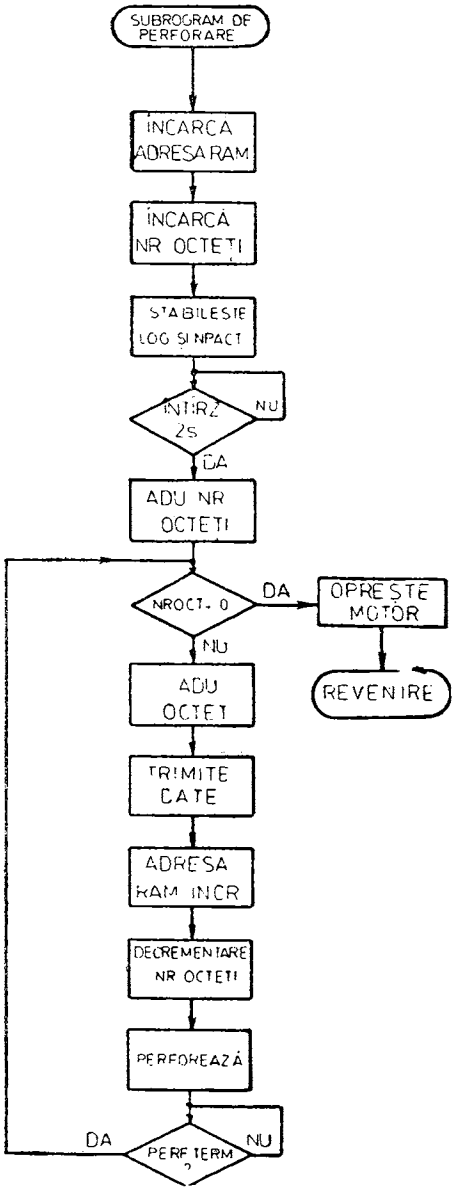


Fig. 3.27. Organigrama subprogramului de perforare

cu semnalele *ADPER* și *I/OW*. Dacă *CP* rămîne în continuare 1 logic de la o perforare la alta, viteza de lucru este maximă.

În interfața din fig. 3.26 modul de lucru adoptat cere ca fiecare caracter să fie însoțit de o comandă de perforare proprie. Comanda de perforare se memorează într-un bistabil de tip D prin semnalele *I/OW* și $\overline{ADRCP}$. *CP* este un semnal repus pe 1 logic prin semnalul *RBPER*. Starea fanionului SR poate fi testată printr-o instrucție de intrare avînd codul de selecție al portului 90_{16}. Nivelul semnalului *OPAR* este forțat la 1 logic pentru inhibarea controlului de paritate.

3.4.3. SUBPROGRAMUL DE PERFORARE

Perforarea unui număr de octeți *NROCT* aflați în memorie într-o zonă de memorie care începe la adresa *ADROCT* se realizează de către subprogramul de perforare (fig. 3.27), (Tabelul 3.17).

Subprogramul începe prin activarea perforatorului și așteptarea celor 2 secunde necesare pornirii motorului. Odată cu activarea se stabilește și semnalul *LOG*. Nivelele logice ale celor două semnale *NPACT* și *LOG* se încarcă prin instrucția *OUT* 9*C*. Apoi se încarcă în *HL* adresa de memorie de unde se trimit octeții de perforat și în continuare numărul de octeți în registrul *B* cu rol de contor.

Declanșarea perforării are loc prin instrucția *OUT* 94, după care se citește continuu și se testează starea fanionului *SR* prin instrucția *IN* 90 și

Tabelul 3.17

Subprogram de perforare pentru P50

```
;  SUBPROGRAMUL PERMITE PERFORAREA UNUI NUMĂR DE OCTEŢI AFLAŢI
;  ÎN MEMORIE (MAX. 256)
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   ADR     EQU     20A4H     ;  ADRESA ZONEI DE MEMORIE DE UNDE SE
                             ;  EXTRAG OCTEŢII
   NROCT   EQU     20AH6     ;  LOCAŢIA UNDE SE AFLĂ NUMĂRUL DE
                             ;  OCTEŢI
   LOGNP   EQU     9CH       ;  ADRESA BISTABILELOR LOG ŞI NPACT
   PORT    EQU     94H       ;  PORTUL DE IEŞIRE DATE
   CP      EQU     98H       ;  ADRESA BISTABILULUI CP
   FAP     EQU     90H       ;  ADRESA PORTULUI DE INTRARE FANIOANE
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   PERF:   MVI A,  03H       ;  SE ÎNCARCĂ ACUMULATORUL PT. LOG=0
                             ;  ŞI NPACT=0
           OUT     LOGNP     ;  SE STABILEŞTE LOG=0 ŞI NPACT=0.
                             ;  S-A PORNIT MOTORUL PERFORATORULUI
           CALL    SEC2      ;  SE APELEAZĂ SUBPROGRAMUL PENTRU
                             ;  INTERVALUL DE 2 SECUNDE
           LHLD    ADR2      ;  SE ÎNCARCĂ ÎN HL ADRESA ZONEI RAM
           LDA     NROCT     ;  SE ÎNCARCĂ NUMĂRUL DE OCTEŢI ÎN
           MOV     B, A      ;  CONTORUL B
           CPI     00H       ;  NR. OCTEŢI=0 ?
   BUCLA:  JZ      FIN       ;  REVENIRE DACĂ NROCT=0
           MOV     A,M       ;  SE ADUCE OCTETUL DE PERFORAT
           OUT     PORT      ;  SE EXTRAGE OCTETUL
           OUT     CP        ;  SE GENEREAZĂ COMANDA DE PERFORARE
           INX     H         ;  SE MĂREŞTE ADRESA RAM
    TEST:  IN      FAN       ;  SE ÎNCARCĂ FANIONUL
           ANI     01H       ;  SE APLICĂ MASCA
           JNZ     TEST      ;  S-A TERMINAT PERFORAREA ?
           DCR     B         ;  DECREMENTEAZĂ CONTORUL DE OCTEŢI
           JMP     BUCLA
    FIN:   MVI     A, 00H    ;  SE ÎNCARCĂ ACUMULATORUL PENTRU
                             ;  DEZACTIVARE
           OUT     LOGNP     ;  SE OPREŞTE MOTORUL
           RET
```

JNZ TEST. Dacă bitul A_4 al acumulatorului are valoarea 1 logic se poate trece la perforarea următorului caracter. Perforarea continuă pînă cînd contorul de octeți, registrul B ajunge la zero, moment în care se oprește motorul perforatorului.

3.5. INTERFAȚA SERIE ASINCRONĂ REALIZATĂ PRIN PROGRAM

3.5.1. INTRODUCERE

Pentru conectarea la microprocesor a unui echipament care poate schimba date în regim de transmisie serie asincronă, se poate avea în vedere o soluție fără circuit de tip UART deci cu reducerea substanțială a complexității circuitelor și trecerea funcțiilor acestora pe seama microprocesorului și a unor programe adecvate.

O astfel de soluție pentru realizarea interfeței cu DAF 1001 folosind tehnica proiectării logice programate se va descrie în continuare [9].

3.5.2. ANALIZA FUNCȚIONALĂ

În cadrul transmisiei asincrone biții de date sînt emiși serie, în timp ce microprocesorul operează cu 8 bit în paralel, de aceea trebuie asigurată conversia paralel-serie (pentru emisie) și cea serie-paralelă (pentru recepție). Analiza funcțională în cazul recepției arată că sînt necesare următoarele operații în cazul recepției *a*) detectarea bitului de START ; *b*) generarea întîrzierii de 4,5 ms pentru a găsi mijlocul duratei bitului START, a duratei de 9 ms pentru eșantionarea biților de date și 18 ms pentru durata celor doi biți de *STOP* ; *c*) numărarea biților ; *d*) recepția biților de date și asamblarea lor în acumulator.

Aceeași analiză pentru emisie conduce la operațiile de mai jos :
a) generarea bitului de *START* ;
b) generarea biților de *STOP* ;
c) generarea întîrzierilor de 9 și 18 ms ;
d) emiterea biților caracterului (7 biți de date + bitul de paritate);
e) numărarea biților.

3.5.3. CIRCUITELE INTERFEȚEI

Partea de circuite a interfeței este extrem de simplă (fig. 3.28). Portul de ieșire spre *DAF* 1001 emite doar un bit la un moment dat, prin urmare cuprinde un singur bistabil de ieșire conectat la linia de date D_0 și sincronizat prin semnalele *I/OW*; *SELDAF*, ultimul obținut prin decodificarea codului $\overline{A_{13}}.\overline{A_{12}}.A_{11}.A_{10}$ de pe liniile de adresă. Pentru recepție, un amplificator trei stări *SN* 74126 selectat de *I/OR.SELDAF* permite citirea stării liniei de legătură cu *DAF*-ul și introducerea valorii logice corespunzătoare în acumulator pe poziția A_0.

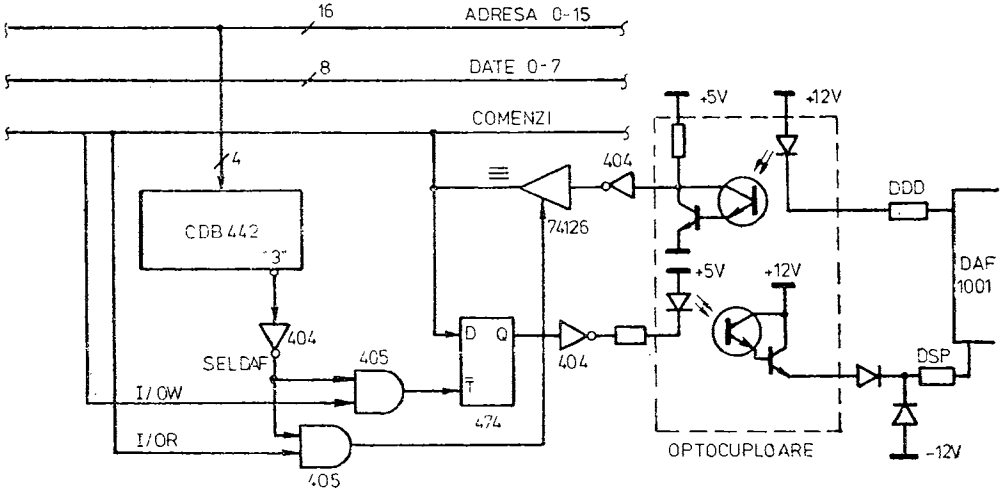


Fig. 3.28. Circuitele de interfață pentru conectarea DAF la microcalculator

3.5.4. SUBPROGRAMUL DAFIN

Recepția este realizată de subprogramul *DAFIN* care permite asamblarea în acumulator a biților recepționați sub forma unui caracter de 8 bit. Fig. 3.29 și tabelul 3.18, înfățișează ordinograma și programul *DAFIN*. După apelare subprogramul transmite caracterul recepționat prin registrul acumulator. Primele trei instrucții ale subprogramului asigură testarea liniei de recepție pentru sesizarea semnalului *START*. Dacă starea liniei devine zero logic, se apelează subprogramul pentru generarea întîrzierii de 4,5 ms și apoi a celui de 9 ms ajungînd astfel pe mijlocul bitului de date B_1 al caracterului emis de *DAF*. Între aceste două întîrzieri se inițializează registrul *D* cu valoarea 1000 000

Tabelul 3.18.

Subprogramul de recepție DAFIN

```
; SUBPROGRAMUL DAFIN POATE RECEPȚIONA UN CARACTER TRANSMIS
; SERIE ASINCRON ÎN FORMATUL 11/8 CU VITEZA 110 BD. LA SFÎRȘIT CARACTERUL
; RECEPȚIONAT SE AFLĂ ÎN ACUMULATOR
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
DAF     EQU     A200H     ; ADRESA DAȚE SERIE DE LA DAF
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
DAFIN : IN      DAFR      ; SE INTRODUCE STAREA LINIEI DE DATE
        RRC               ; SE DEPLASEAZĂ ÎN CARRY
        JC      DAFIN     ; DACĂ NU E BITUL START SALT LA DAFIN
        CALL    TIMP      ; INTERVAL DE 4,5 MS
        MVI     D, 80H    ; SE INIȚIALIZEAZĂ REGISTRUL D
RECPT : CALL    TIMP1     ; INTERVAL DE 9 MS
        IN      DAFR      ; SE INTRODUCE STAREA LINIEI DE DATE
        RAR               ; SE DEPLASEAZĂ BITUL ÎN CARRY
        MOV     A,D       ; SE TRANSFERĂ D ÎN A
        RAR               ; SE ATAȘEAZĂ BITUL RECEPȚIONAT
        MOV     D, A      ; SE DEPUNE IN D
        JNC     RECPT     ; DACĂ CARRY=-0 SE CONTINUĂ
        CALL    TIMP2     ; INTERVAL DE 18 MS
        RET
SUBPROGRAME DE REALIZAREA INTERVALELOR DE 4,5 9 și 18 MS
TIMP2 : CALL    TIMP1     ; SE APELEAZĂ RUTINA PENTRU 9MS LA
                          ; REVENIRE SE MAI SCURG 9 MS
TIMP1 : CALL    TIMP      ; SE APELEAZĂ RUTINA PENTRU 4.5 MS.
                          ; LA REVENIRE SE SCURG ALTE 4,5 MS.
TIMP :  XTHL              ; SE CONSUMĂ TIMP ÎN TOTAL
        XTHL              ; 16×2=32 STĂRI
        DCR     B         ; SE DECREMENTEAZĂ REG. B.
        JNZ     TIMP      ; DACĂ B=0 SE REVINE ÎN PROGRAMUL
        RET               ; APELANT
```

(80_{16}). Prin citirea portului de intrare valoarea logică a stării liniei ajunge în acumulator (A_0) și după executarea unei deplasări la dreapta cu participarea bistabilului de condiție *CARRY*, în acesta din urmă.

Prin transferul conținutului lui *D* în acumulator și o nouă deplasare la dreapta se aduce bitul de date în A_7 și zero logic în *CARRY*. Reluarea procesului descris permite asamblarea bit cu bit a caracterului emis serie. Terminarea recepției este semnalată de sosirea bitului de 1 logic aflat inițial în poziția c.m.s. în poziția c.m.p.s. a acumulatorului și testarea lui în *CARRY*.

Pentru scurgerea timpului corespunzător celor doi biți de *STOP* se mai generează o întîrziere de 18 ms. La revenirea în programul principal caracterul recepționat se află în acumulator.

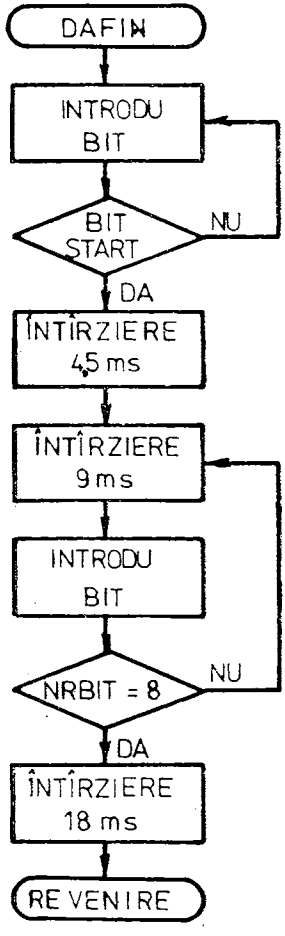


Fig. 3.29. Organigrama subprogramului DAFIN

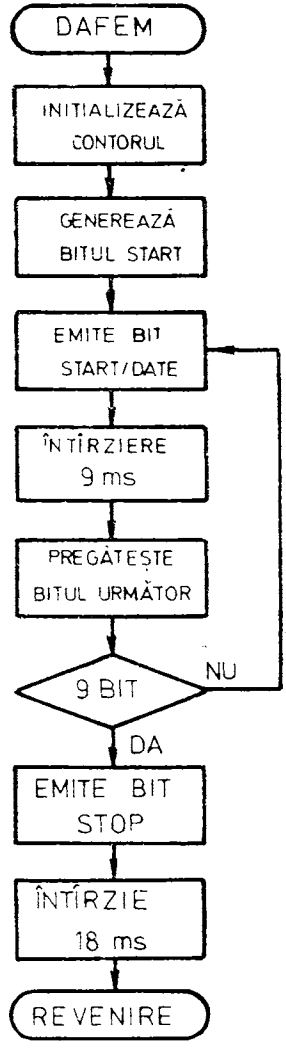


Fig. 3.30. Organigrama subprogramului DAFEM

3.5.5. **SUBPROGRAMUL** DAFEM

Acest subprogram este destinat emisiei unui caracter în formatul 11/8 cu viteza de 110 Bd. Caracterul de emis este transmis de către programul principal spre subprogramul *DAFEM* prin registrul acumulator. Organigrama și lista instrucțiilor sînt arătate în fig. 3.30 și tabelul 3.19. La începutul subprogramului se inițializează registrul *C* care va servi pentru numărarea biților emiși.

Printr-o instrucțiune *ANA A* se aduce la zero bistabilul *CARRY*. O deplasare la stînga aduce acest 0 logic în poziția A_0 a acumulatorului. Acest zero va fi emis ca bit de *START* printr-o instrucțiune de memorare în bistabilul de ieșire. După întîrzierea de 9 ms se continuă generarea biților de date prin deplasări la dreapta, memorarea bistabilului de ieșire, incrementarea lui *C* pînă la contor zero, adică după emiterea a 9 biți în total. Emisia încetează prin înscrierea unui 1 logic în bistabilul de ieșire și generarea unei întîrzieri de 18 ms echivalentă cu prelucrarea celor 2 biți de *STOP*. Trebuie remarcat modul de generare a întîrzierilor prin intermediul subprogramelor *TIMP*, *TIMP* 1 și *TIMP* 2, folosite atît de *DAFIN* cît și de *DAFEM*. Întîrzierea de 4,5 ms prin subprogramul *DLY* se bazează pe numărarea în registrul B și executarea perechii de instrucțiuni *XTHL* care nu produce alt efect decît consumarea a 38 de stări [10]. Alte 13 stări necesare decrementării și testării ne conduc

Tabelul 3.19.

Subprogramul de emisie DAFEM

```
;  SUBPROGRAMUL DAFEM POATE EMITE UN CARACTER SERIE ÎN TRANSMISIE
;  ASINCRONA FORMATUL 11/8 CU VITEZA 110 BD. CARACTERUL CE SE EMITE
;  SE AFLĂ LA ÎNCEPUT ÎN ACUMULATOR
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   DAF     EQU      A200H      ADRESA DATE SERIE SPRE DAF
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   DAFEM : MVI      C, 0F7H    ;  SE INIȚIALIZEAZĂ CONTORUL REALIZAT
                               ;  ÎN C
           ANA      A          ;  SE ADUCE CARRY LA ZERO
           RAL                 ;  SE PREGĂTEȘTE BITUL DE START ÎN ACU-
                               ;  MULATOR
   EMIS    OUT      DAF        :  SE EMITE BITUL PREGĂTIT
           CALL     TIMP1      ;  INTERVAL DE 9 MS
           RAR                 ;  SE ADUCE UN NOU BIT PENTRU EMISIE
           INR C               ;  SE INCREMENTEAZĂ C
           JNZ      EMIS       ;  NOUĂ BIȚI EMIȘI 2 NU, SALT LA EMIS
           MVI A,   01H        ;  SE PREGĂTEȘTE BITUL STOP
           OUT      DAF        ;  SE EMITE BITUL STOP
           CALL     TIMP2         INTERVAL DE 18 MS
           RET
```

la un total de 51 de stări. Pentru generarea unei întîrzieri de 4,5 ms rezultă o valoare inițială cu care trebuie încărcat registrul B : $N=TIMP/51.T_{stare}$ care pentru valorile 4,5 ms și $T_{stare}=0{,}5$ microsecunde dă $N=174$.

Pentru generarea întîrzierilor de 9 ms și 18 ms care reprezintă fiecare dublul, respectiv cvadruplul celei de 4,5 ms, au fost plasate două instrucții *CALL*. Acest lucru este de natură să permită executarea de 2 ori sau de 4 ori a subprogramului TIMP.

Există un dezavantaj în implementarea unei astfel de interfețe realizate prin program prin ocuparea unității centrale de microprocesor pe întreaga durată a emisiei sau recepției. Printre avantaje sînt posibilitatea adaptării ușoare la alte regimuri de viteză sau format schimbînd programele și circuitele de interfață reduse ca număr.

3.6. ÎNREGISTRAREA DATELOR BINARE PE CASETOFON

3.6.1. INTRODUCERE

O soluție avantajoasă pentru înregistrarea programelor și încărcarea acestora în memoria microcalculatorului atunci cînd e necesar o constituie utilizarea unui casetofon obișnuit. În vederea înregistrării informațiilor binare nu trebuiesc aduse modificări schemei casetofonului dar e nevoie de o interfață care va fi descrisă în cele ce urmează [11].

Metodele obișnuite de înregistrare a datelor numerice binare pe un suport magnetic cum sînt, metoda cu reîntoarcere la zero (*RZ*), fără reîntoarcere la zero (*NRZ*), modulației în fază (*MF*) nu sînt potrivite pentru casetofon, deoarece schema electronică a acestuia are caracteristica de frecvență limitată la domeniul frecvențelor audio, insuficient pentru a înregistra sau reda impulsuri.

O alternativă este codificarea în frecvență (fig. 3.31) în cadrul căreia „1" logic este reprezentat printr-un semnal sinusoidal (ton) cu frecvența F_1, iar pentru

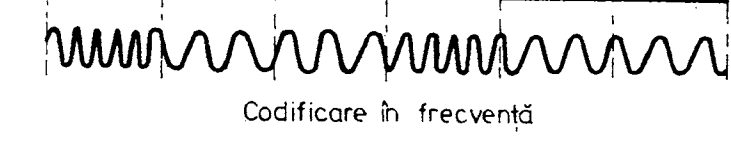


Fig. 3.31. Codificarea în frecvență a semnalelor binare

„0" logic se ia o altă frecvență F_2. În consecință datele aflate în memoria microcalculatorului, sub forma a 8 bit pe cuvînt (octet), vor fi transmise serie, ca undă binară și după codificarea în frecvență vor fi înregistrate pe casetă.

3.6.2. DESCRIEREA INTERFEȚEI

Schema de principiu a interfeței este arătată în fig. 3.32. Datele binare sînt preluate de la ieșirea serie pentru transmisie asincronă (*UART*) a microcalculatorului aceeași ca pentru legătura cu teleimprimatorul sau DAF-ul. Viteza de transmitere este de 110 Bd (10 caractere/s) în formatul 11/8 adică 8 biți utili încadrați de start și doi bit de stop. Succesiunea de nivele binare 1 sau 0 logic primite de la ieșirea serie este transformată de blocul MODULATOR pilotat de GENERATORUL DE TACT cu cristal de cuarț, într-o undă rectangulară în care se succed frecvențele 2 600 Hz („1") și 2080 Hz („0"). Prin intermediul FILTRULUI DE IEȘIRE se obține un semnal apropiat de forma sinusoidală cu amplitudinea 300—400 mV care se trimite la borna de înregistrare a casetofonului. Pentru redare semnalul cules de la borna de cască cu care este prevăzut casetofonul intră în LIMITATORUL DE AMPLITUDINE, apoi într-un AMPLIFICATOR-FORMATOR și în cele din urmă în DISCRIMINATORUL DE FRECVENȚĂ care reface unda binară inițială din timpul înregistrării. Deoarece înregistrarea și redarea nu se fac simultan, pentru modulare și demodulare se pot utiliza elemente din schema în comun, iar fixarea regimului de lucru are loc prin poziționarea unui comutator acționat manual. Schema completă a interfeței pentru casetofon este dată în fig. 3.33. Numărătorul cu 9 ranguri binare comandat cu impulsuri de tact cu frecvența 1 MHz de la generatorul de impulsuri de tact determină durata unei alternanțe pentru frecvențele F_1 sau F_2 prin aceea că numără pînă la N_1

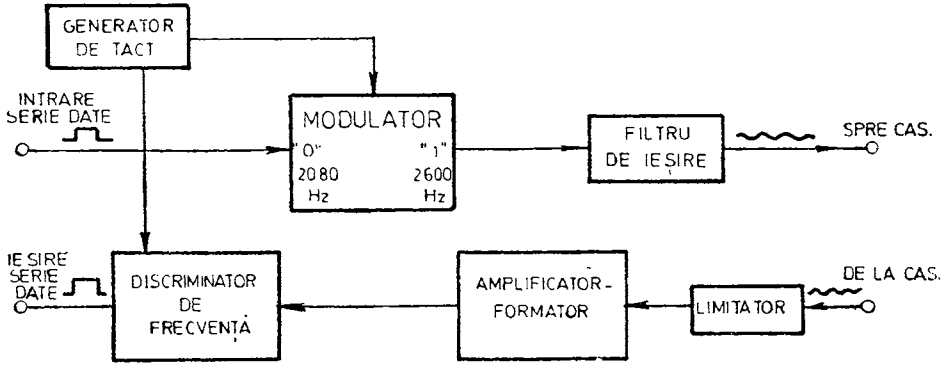


Fig. 3.32. Schema de principiu a interfeței pentru casetofon

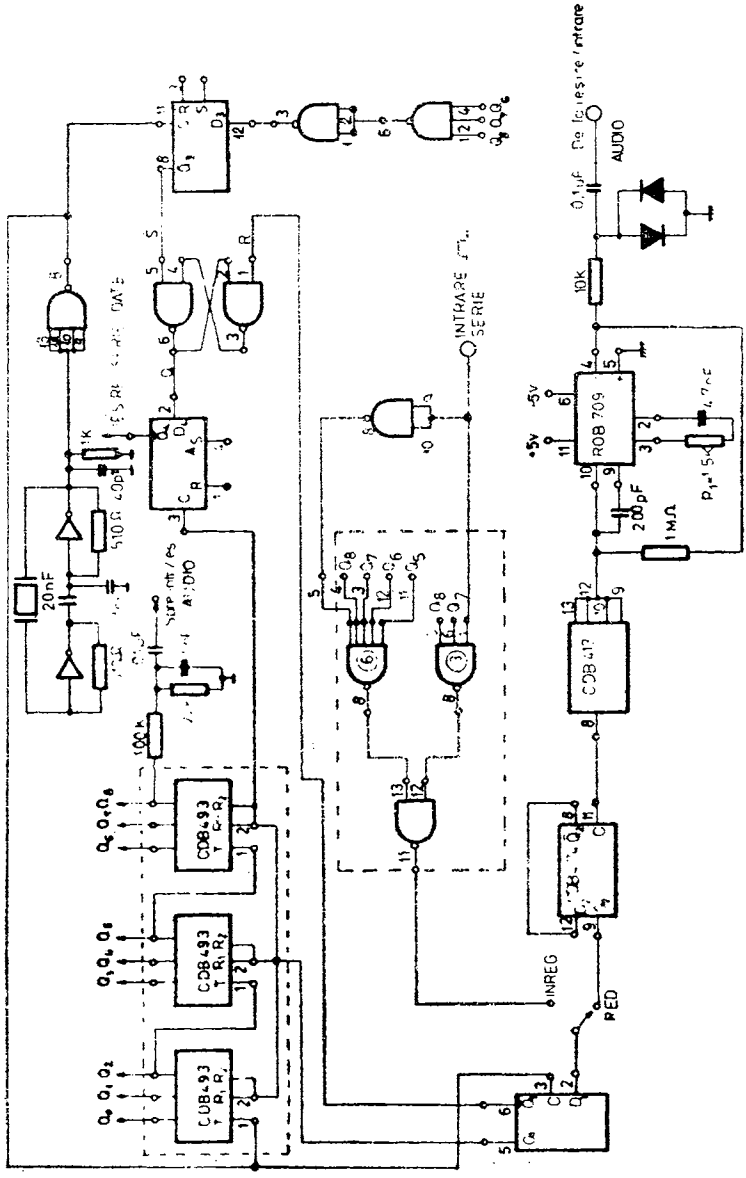


Fig. 3.33. Schema logică a interfeței

sau pînă la N_2 două valori calculate funcție de F_1 și F_2. În vederea demodulării același numărător va fi folosit pentru discriminarea frecvențelor după cum se va depăși sau nu valoarea de prag N_p. Cele două regimuri de lucru: înregistrare și redare, vor fi descrise separat.

3.6.3. ÎNREGISTRAREA

Dacă comutatorul ÎNREG/REDARE se află pe poziția ÎNREG și dacă la intrarea de date se aplică o succesiune de nivele binare, vor fi validate una din porțile 3 sau 6. Pentru 1 logic și starea $N_1=384=(2_8+2_7)$ corespunzătoare frecvenței 2 600 Hz, poarta 3 va furniza la ieșire un zero logic și un 1 logic la intrarea bistabilului D_1. Dimpotrivă dacă la intrarea de date va fi un nivel 0 logic se decodifică valoarea $N_2=480$ de către poarta 6. Valoarea $480=(2_8+2_7+2_6+2_5)$ corespunde frecvenței 2080 Hz. Indiferent dacă e vorba de un 1 sau de un 0 logic la intrarea bistabilului D_1 va apare mai devreme sau mai tîrziu un 1 logic. La primul impuls de tact ieșirea lui D_1 va deveni 1 și întrucît de aici se comandă intrările R_1 și R_2 ale numărătoarelor *CDB* 493 acestea vor fi aduse la zero. După aducerea la zero a numărătoarelor dispare condiția care a generat 1 la intrarea bistabilului D_1, prin urmare se reia numărarea. Rezultatul va fi că numărătorul va parcurge de mai multe ori stările de la 0 la 384 sau 480 în funcție de valoarea logică a datelor la intrare și de durata acestora. Numărările repetate conduc la punerea pe zero și pe unu a

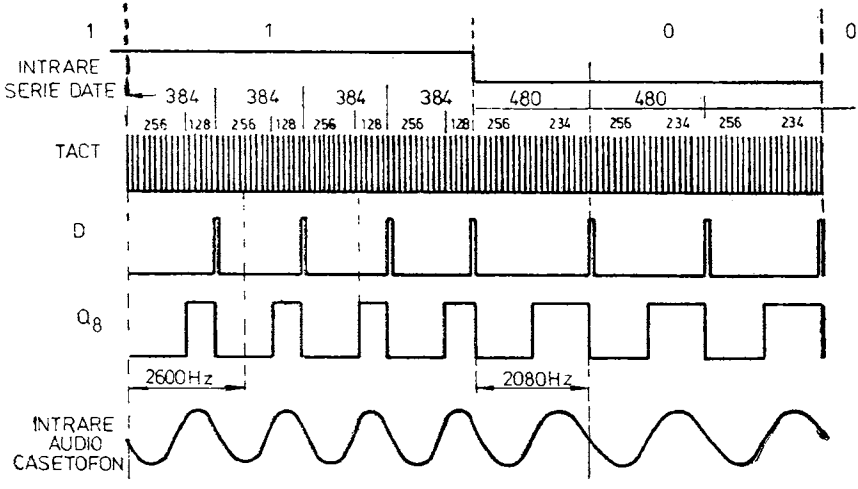


Fig. 3.34. Diagrama de timp la ÎNREGISTRARE

bistabilului Q_8 cu perioade diferite $1/F_1$ sau $1/F_2$. Unda binară generată de la ieșirea Q_8 este filtrată pentru eliminarea armonicilor cu frecvențe din afara domeniului audio. Diagrama de timp pentru semnalele descrise mai sus apare în fig. 3.34.

3.6.4. REDAREA

Semnalul cules la ieșirea pentru cască a casetofonului este limitat în amplitudine de cele două diode de la intrarea în lanțul de redare. După limitarea ambelor alternanțe semnalul este aplicat unui amplificator care îi va ridica nivelul de amplitudine corespunzător intrării unui circuit bistabil basculant Schmitt, *CDB* 413. Circuitul bistabil Schmitt va forma fronturile semnalului corespunzător normelor TTL și apoi frecvența acestuia va fi divizată cu doi de către bistabilul D_2, conectat ca bistabil de tipul „*t*". Dacă comutatorul de regim se află pe poziția REDARE atunci semnalul ajunge la intrarea bistabilului D_1 al cărui rol este de a sincroniza cu tactul semnalul util. În cazul unei frecvențe de 2 600 Hz numărătorul va reuși să atingă valoarea 384 iar pentru frecvența 2080 Hz valoarea 480. Valoarea pentru discriminarea celor două frecvențe a fost aleasă de 448. Funcție de această valoare prag se va înscrie 1 sau 0 în bistabilul D_3 și apoi în D_4. Diagrama de timp a semnalelor pentru regimul de redare este dată în fig. 3.35. Datele serie spre microcalculator sînt disponibile la ieșirea $\overline{Q}$ a bistabilului D_4.

În mod obișnuit viteza de derulare a benzii din casetele magnetice este de 4,75 cm/s. Pentru o lungime de bandă a unei casete de 60 de minute 17 000 cm și viteza de transmisie 110 bit/s se obține o densitate de înregistrare de :

$d = 110/0{,}475 = 2{,}3$ bit/mm și o capacitate de înregistrare

$NR.BIT = 170\,000 . 2{,}3 = 396\,000$ bit.

Deoarece în transmisia serie asincronă în formatul 8/11 numai cei 8 biți sînt biții de date, dar se înregistrează și bitul de start și cei doi biți de stop, capacitatea de înregistrare măsurată în octeți este :

$$NR\ .OCTEȚI = 396\,000/11 = 36\,000 \text{ octeți.}$$

Conectarea interfeței pentru casetofon la microcalculator se face întocmai ca pentru *DAF* 1001 sau orice alt echipament avînd transmisie serie asincronă.

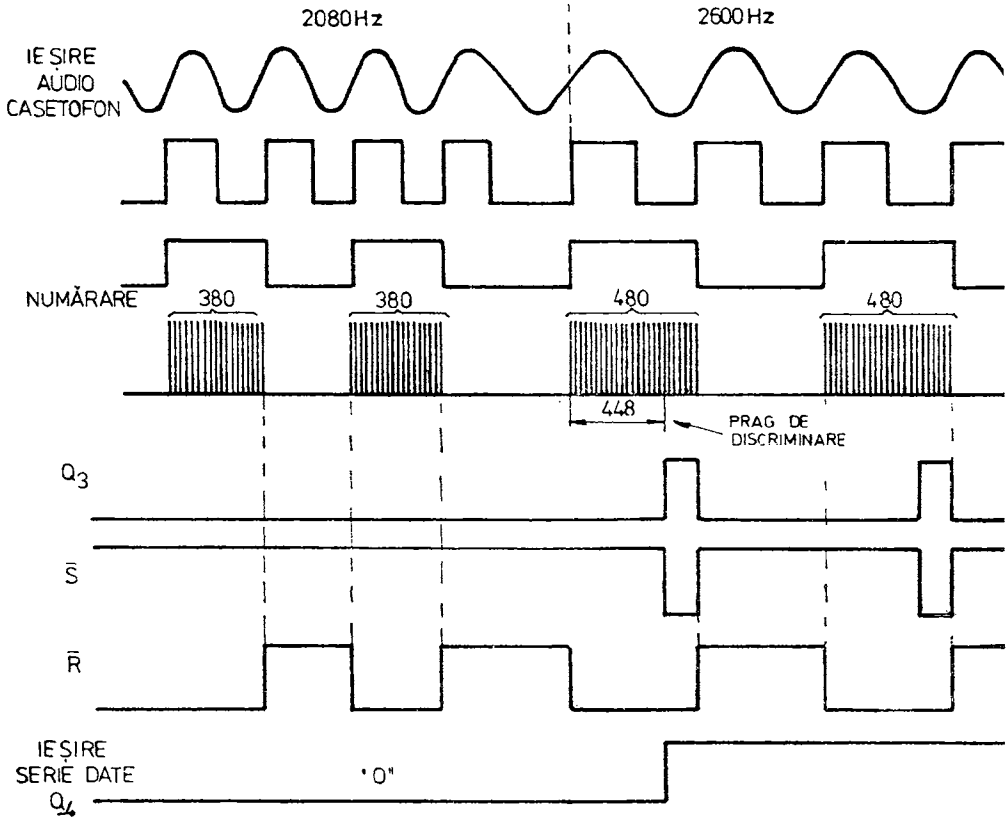


Fig. 3.35. Diagrama de timp la REDARE

3.7. SISTEM PENTRU ÎNTRERUPERI MULTIPLE

3.7.1. INTRODUCERE

Efectuarea corectă a transferului de date cu un periferic nu poate avea loc decît dacă microprocesorul cunoaște momentul în care acesta este pregătit pentru transfer. Se știe însă că testarea acestui moment prin program consumă timp și microprocesorul este încetinit considerabil. Apare deci mai rațional ca perifericul însuși să-și semnaleze starea în care e disponibil transferului de date. Acesta este principiul cererii de întrerupere.

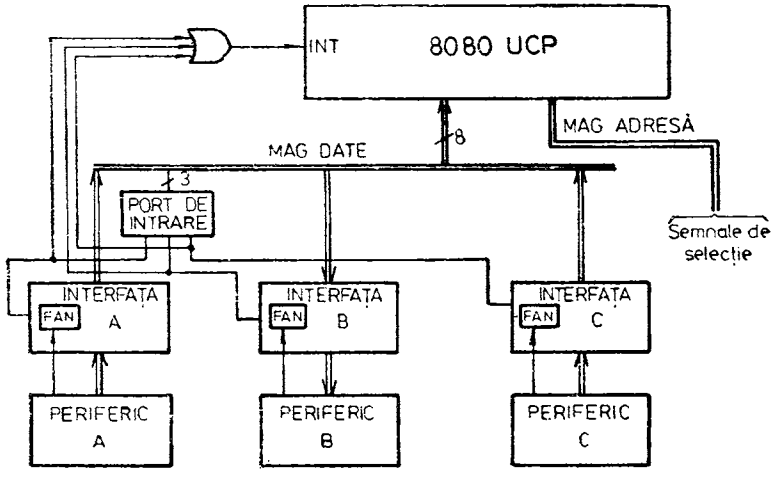


Fig. 3.36. Metoda întreruperilor prin scrutarea perifericelor (polling)

Microprocesorul 8080 nu posedă decît o singură intrare de întreruperе *INT*. Dacă sînt mai multe periferice, așa cum se întîmplă frecvent, liniile de cerere de întrerupere ale acestora pot fi reunite printr-o funcție SAU a cărei ieșire este conectată la intrarea *INT*.

Se pune acum o dublă problemă : 1) microprocesorul trebuie să identifice perifericul care a lansat întreruperea ; 2) în cazul mai multor cereri de întrerupere apărute simultan este necesar ca microprocesorul să știe în ce ordine de prioritate să servească perifericele [12], [13].

Există două metode curent folosite, „scrutarea" prin program (polling) și întreruperea dirijată (vectored interrupt). În cadrul primei metode (fig. 3.36) fiecare interfață de periferic dispune de un bistabil fanion validat de cererea de întrerupere. Procedura de întrerupere multiplă prin scrutarea perifericelor se desfășoară ca în fig. 3.37. Imediat după acceptarea întreruperii se salvează conținutul registrelor *A* și *F*, *BC*, *DE*, *HL* în stivă și prin programul de „scrutare" se testează succesiv fiecare fanion. Ordinea de prioritate este chiar ordinea în care se testează fanioanele. În fig. 3.37 de exemplu, perifericul *A* este cel mai prioritar. Se poate interveni în această ordine de prioritate dacă în programul de serviciu *A* se inserează o instrucție de autorizare a întreruperilor *EI*. Perifericul *A* își pierde prioritatea deoarece va putea fi întrerupt de un alt periferic. Procedura este destul de simplă și este deseori folosită, dar are și un dezavantaj că se consumă timp mai ales în cazul defavorabil cînd trebuie testate toate perifericele înainte de a detecta perifericul în cauză.

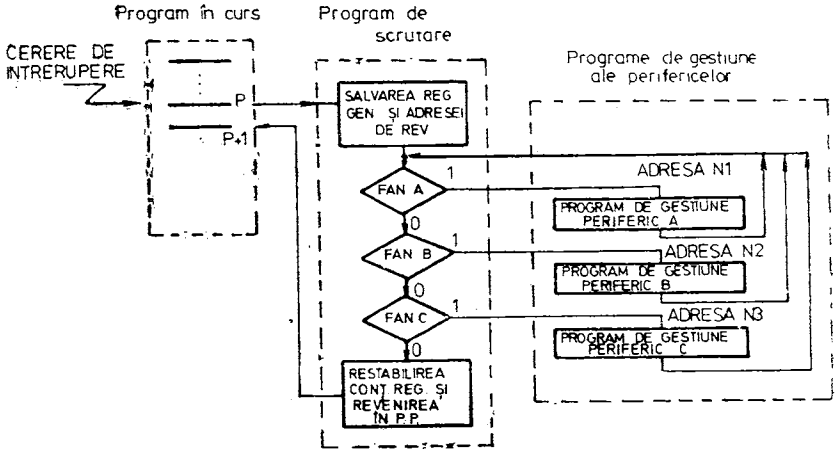


Fig. 3.37. Procedura de întrerupere multiplă prin scrutarea periferice lor (polling)

Procedura de întreruperi dirijate prevede pentru fiecare periferic un program de serviciu depus în memorie la o adresă dată : programul perifericului A se află depus în memorie de la adresa N_1, acela pentru perifericul B la adresa N_2. Particularitatea metodei este că aceste adrese N_1 și N_2 se află ele însele depuse într-o porțiune a memoriei, la adrese desemnate de vectori de întrerupere. În cursul unei întreruperi microprocesorul primește valoarea vectorului de întrerupere, face legătura cu adresa vectorului cerut și sare la programul de serviciu cerut. Se recunoaște desigur în acest mecanism principiul adresării indirecte. Schema prin care se realizează (fig. 3.38) are un caracter care s-a generalizat în așa măsură încît există circuite speciale destinate acestor funcțiuni ca 8214 și 8259. Perifericul trimite întreruperea atît spre microprocesor cît și spre circuitul de rezolvarea priorităților. Codificatorul de prioritate generează în funcție de prioritatea perifericului care a cerut întreruperea, o adresă vector. Această adresă poate fi dată sub forma instrucției *RST N* sau *CALL Adr.* În primul caz microprocesorul se branșează la adresa specificată de vector iar în al doilea caz microprocesorul mai execută două cicluri de întrerupere în care se introduc cei doi octeți ce formează adresa de salt. În concluzie această procedură permite „dirijarea" microprocesorului direct spre programul de serviciu al perifericului prioritar.

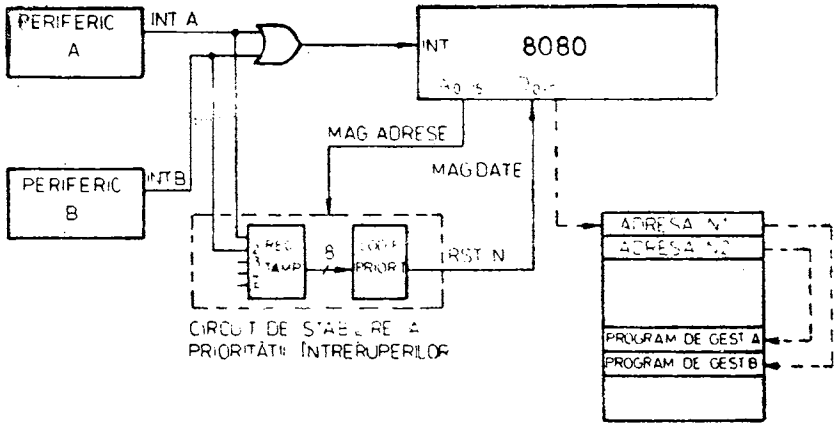


Fig. 3.38. Metoda întreruperilor dirijate

3.7.2. SCHEMA CIRCUITELOR DE ÎNTRERUPERE

Pentru microsistemul de dezvoltare s-a ales o procedură de realizare a întreruperilor multiple care reprezintă o combinație între cele două metode înainte prezentate.

Schema circuitelor de întrerupere (fig. 3.39) primește cererile de întrerupere de la tastatura *TO-4* (*INTROP*), de la lector ($\overline{INTL}$), de la perforator ($\overline{INTP}$) și de la *DAF* ($\overline{INTE}$, $\overline{INTR}$). Interfața cu *DAF* 1001 se presupune că e realizată cu circuitul 8251 *USART*. Distincția între *INTROP* și celelalte semnale se face prin metoda întreruperilor dirijate. Semnalul *INTROP* are prioritatea cea mai mare. Între celelalte periferice prioritatea se rezolvă prin metoda scrutării. Toate semnalele de întrerupere lansate de periferice sînt memorate în bistabile și sînt reunite printr-o funcție *SAU* într-un singur semnal care comandă intrarea *INT* a microprocesorului. Toate semnalele de întrerupere cu excepția lui *INTROP* au fost grupate printr-o funcție *SAU* în semnalul *INTPER*. Presupunînd că există cel puțin o întrerupere de periferic, INTPER va fi 1 logic și prin urmare în momentul executării ciclului de întrerupere D_3 va fi 1 logic. Se creează astfel vectorul de dirijare *RST* 7 deoarece celelalte linii de date sînt toate pe 1 logic. Prin *RST* 7 microprocesorul se branșează pe subprogramele de gestiune ale perifericelor care se află în afara MONITORULUI.

Dimpotrivă, dacă apare o întrerupere *INTROP* de la tastatură linia D_3 devine 0 logic în momentul executării ciclului de întrerupere, adică se gene-

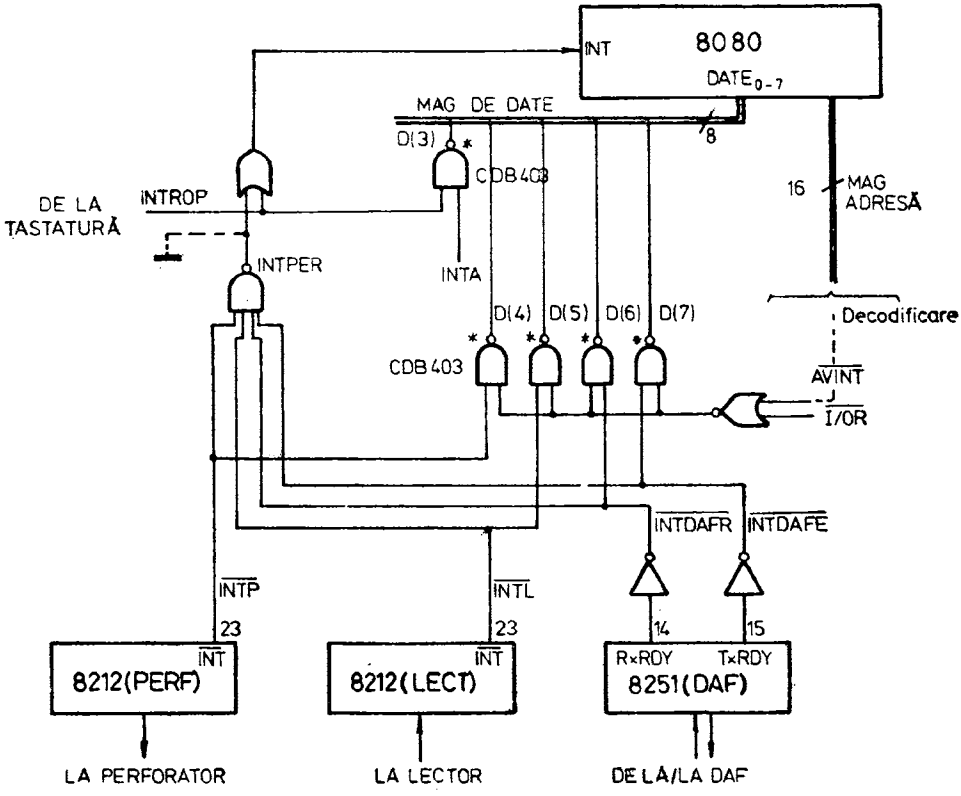


Fig. 3.39. Schema logică a circuitelor de întrerupere

reză vectorul *RST6*. Astfel microprocesorul se branşează pe adresa 0030_{16} iar de aici la programul de tratare a întreruperii de la tastatură *TO-4* aflat în MONITOR.

Fanioanele din interfeţe pot fi citite printr-o instrucţie de intrare care generează semnalul $\overline{AVINT.I/OR}$ pentru autorizarea introducerii stărilor logice ale semnalelor *INTL, INTP, INTE* şi *INTR* pe magistrala de date şi apoi în acumulator.

După iniţializarea prin *RESET* toate fanioanele care emit semnalele de întrerupere trebuie să ajungă în starea în care nu se cere nici o întrerupere.

Tabelul 3.20.

Subprogramul de tratare a întreruperilor de periferice

```
;; ACEST SUBPROGRAM DETERMINĂ PERIFERICELE CARE AU LANSAT ÎNTRE-
   RUPERI ACORDÎND PRIORITĂŢILE ÎN ORDINEA : DAF (RECEPŢIE), DAF (EMI-
   SIE), LECTOR, PERFORATOR.
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
AVÎNT   EQU     090H        ; ADRESA PORTULUI PENTRU INTRAREA
                            ; FANIOANE DE ÎNTRERUPERE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        ORG     0038H
        CALL    ANPER       ; SE APELEAZĂ SUBPR. DE ANALIZĂ A INTR.
        EI                  ; SE AUTORIZEAZĂ ÎNTRERUPERILE
        RET                 ; SE REVINE LA PROGRAMUL ÎNTRERUPT
        ORG     1010H
ANPER : PUSH    PSW         ; SE SALVEAZĂ ÎN STIVA FANIOANE+ACUM.
        PUSH    B           ; SE SALVEAZĂ REG B ŞI C
        PUSH    D           ; SE SALVEAZĂ REG. D ŞI E
        PUSH    H           ; SE SALVEAZĂ REG. H ŞI L
        LDA     AVÎNT       ; SE CITESC STĂRILE FANIOANELOR
        RLC                 ; SE DEPLASEAZĂ LA STÎNGA
        STA     FANI        ; SE DEPUNE ACUM. ÎN LOCAŢIA FANI.
        CC      DAFIN       ; DACĂ CARRY=1, ÎNTRERUP. LA DAFIN
        LDA     FANI        ; SE ÎNCARCĂ FANIOANELE
        RLC                 ; SE DEPLASEAZĂ
        STA     FANI        ; SE DEPUNE ÎN MEMORIE
        CC      DAFEM       ; TESTEAZĂ DACĂ INTRAREA DATEM
        LDA     FANI        ; SE ÎNCARCĂ FANIOANELE
        RLC                 ; SE DEPLASEAZĂ CONT. ACUMUL.
        STA     FANI        ; PENTRU A SE TESTA ALT FANION
        CC      LECT        ; DACĂ CARRY=1, ÎNTRERUP. LECTOR
        LDA     FANI        ; SE READUC FANIOANELE
        RLC                 ; SE DEPLASEAZĂ LA STÎNGA
        CC      PERF        ; SE TESTEAZĂ FAN. INTR. PERFORATOR
        POP     H           ; SE REFACE CONŢINUTUL H ŞI L
        POP     D           ; SE REFACE CONŢINUTUL D ŞI E
        POP     B           ; SE REFACE CONŢINUTUL B ŞI C
        POP     PSW         ; SE REFACE CONŢINUTUL A ŞI F
        RET                 ; REVINE
```

3.7.3. SUBPROGRAMUL DE TRATARE A ÎNTRERUPERILOR MULTIPLE

În cazul întreruperilor cerute de lector, perforator sau *DAF* se execută instrucția RST 7 prin care se ajunge la adresa 0038_{16} și apoi la subprogramul care tratează întreruperile prin scrutare.

Acesta începe prin citirea fanioanelor în acumulator. Urmează (Tab. 3.20) analiza prin program a biților 7, 6, 5 și 4 în această ordine pentru a identifica perifericele care trebuie servite. Se remarcă desigur că prioritatea maximă a fost acordată *DAF*-ului mai exact receptorului din circuitul 8251 care emite cererea de întrerupere din partea operatorului.

Testarea biților acumulatorului se realizează simplu prin deplasări la stînga și salturi condiționate de valoarea logică 1 a bistabilului de condiție *CARRY*. Între două testări conținutul acumulatorului este memorat în *RAM* la adresa *FAN* pentru a relua testarea după operațiile solicitate de un program de gestiune.

Posibilitatea de a lucra prin tehnica întreruperilor sau prin interogare programată constituie unul din avantajele schemei din fig. 3.39. În cazul transferului prin interogare programată linia *INTPER* se forțează la 0 logic excluzînd astfel întreruperile cu excepția uneia singure ; cea a tastaturii *TO*-4.

Prin executarea programului de gestiune se efectuează transferul de date spre microprocesor (sau de la microprocesor) și se anulează cererea de întrerupere.

O trăsătură specifică programului de tratare a întreruperilor prin metoda scrutării este posibilitatea de a trata întreruperile sosite simultan sau chiar întreruperi sosite în timpul deservirii unui periferic mai prioritar de la periferice mai puțin prioritare.

BIBLIOGRAFIE

1. Ed. Wen,; C. Lin,; *Microprocessors Fundamentals & Applications*, IEEE Press, New York, 1976.
2. H. Falk; *Microcomputer software makes its debut*, IEEE Spectrum p. 78—84 Oct. 1974.
3. I. Dancea; *Microprocesoare*, Editura Dacia, Cluj-Napoca, 1979.
4. * * * *Mikroprozessoren*, Elektronik, 1979.
5. * * * *Proceedings of the IEEE*, 1976 June, Special Issue on microprocessor technology and applications.
6. J.A. Titus: C. Titus; P.R. Roney, D.G. Larsen, *Microprocessor Interfacing : Command Decoders*, Computer Design, June 1978, p. 135.

7. * * * *Lectorul de bandă LB-50*, Carte tehnică, IEPER București.
8. * * * *Perforatorul de bandă P-50*, Carte tehnică, IEPER București.
9. R. Zaks, A; Lesea; *Microprocessor interfacing techniques*, SYBEX Publication, 1977, 1978.
10. A. Osborne; *8080 Programming for Logic Design*, SYBEX Publication.
11. R. Stoinescu, *Echipamente periferice și transmiterea datelor*, îndrumător de lucrări, I. P. Timișoara, 1980.
12. R. Lyon-Caen; J. M. Crozet; *Microprocesseurs et microordinateurs*, Masson, 1978.
13. J. Nicoud; *MUBUS*, Microscope, nr. 10, 1978.

CAPITOLUL 4

INTERFEȚE CU ALTE PERIFERICE NUMERICE

4.1. CONECTAREA UNEI TASTATURI FĂRĂ ELECTRONICĂ DE CODARE [1], [2], [3]

În cele ce urmează prin termenul de tastatură se înțelege o mulțime de contacte acționate prin apăsare manuală. Problema principală a utilizării unei tastaturi o constitue identificarea tastei apăsate. Din acest punct de vedere tastaturile pot fi de două feluri : cu codare și fără codare. Tastaturile cu codare dispun de scheme electronice pentru identificarea tastelor apăsate și memorarea informației pînă la apăsarea unei noi taste. Tastaturile fără codare nu dispun de astfel de scheme electronice, identificarea tastelor apăsate fiind realizată prin programe speciale ale monitorului microsistemului.

4.1.1. PROBLEME ALE TASTATURILOR CU CONTACTE MECANICE

Datorită imperfecțiunii contactelor cu acționare mecanică, între acestea apar oscilații atît în timpul închiderii cît și în timpul deschiderii lor (fig. 4.1). Pentru a se evita posibilitatea interpretării acestor oscilații ca informații false este necesară o filtrare a semnalelor de la ieșirile tastaturii, înainte de interpretarea lor. Filtrarea poate fi realizată fie prin circuite electronice speciale fie prin introducerea unor bucle de întîrziere în cadrul programelor de interpretare a informației de la tastatură. Circuitele de filtrare pot fi de tip *R-C* sau circuite „cu zăvorîre" realizate cu porți *ȘI-NU* (fig. 4.2). Ultima soluție prezintă dezavantajul de a necesita pentru fiecare tastă un comutator cu două contacte complementare.

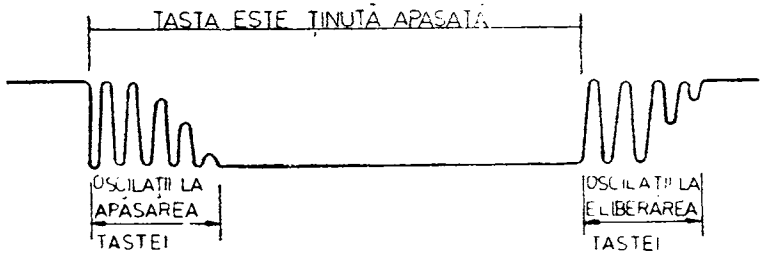


Fig. 4.1. Semnalul obținut la apăsarea unei taste cu contacte mecanice

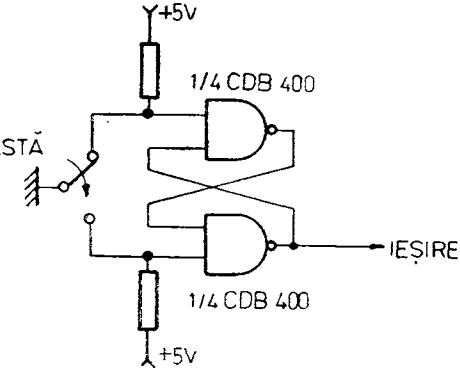


Fig. 4.2. Schemă pentru filtrarea oscilațiilor produse datorită imperfecțiunii contactelor mecanice

Utilizarea circuitelor electronice de filtrare este practic limitată la cazul unui număr mic de taste. În cazul tastaturilor mari se preferă rezolvarea acestor probleme prin subprogramele monitorului.

O altă problemă dificilă care apare în cazul tastaturilor, este cea a posibilităților apăsării simultane, din greșeală a două sau mai multe taste. În fig. 4.3 se prezintă o schemă secvențială asincronă care asigură pe de o parte filtrarea oscilațiilor datorate imperfecțiunii contactelor mecanice și pe de altă parte îndeplinește funcțiunea de „arbitrare" între mai multe eventuale acționări simultane a tastelor. Conflictul care apare este tranșat în favoarea unei singure taste. Schema nu necesită contacte duble pentru taste ca soluția de filtrare din figura precedentă. Utilizînd circuite *ȘI—NU* 74*S*133 schema poate fi extinsă pînă la maximum 13 taste.

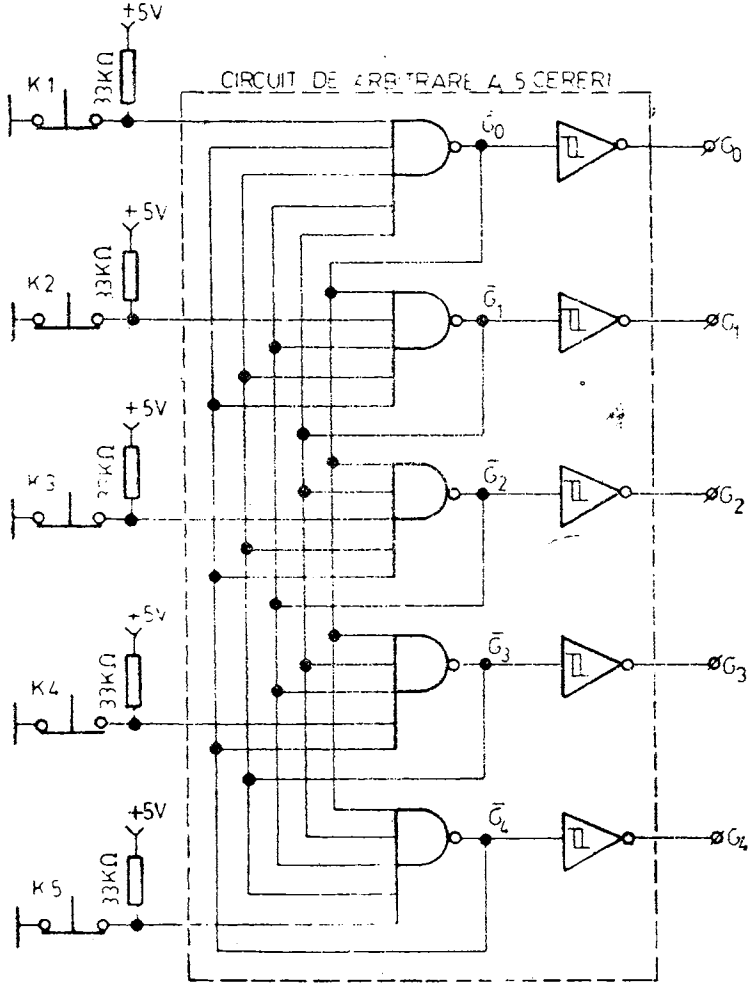


Fig. 4.3. Circuit de filtrare și de arbitrare a oscilațiilor datorită contactelor acționate mecanic

Pe baza schemei din fig. 4.3 se poate concepe un modul de filtrare și arbitrare cu posibilități de interconectare în configurații extinse. În fig. 4.4 se prezintă spre exemplificare schema extinsă pentru 16 taste ($K11$, ..., $K44$). Ieșirile schemei sînt AK_{11}, ..., AK_{14}, ... AK_{41}, ..., AK_{44}.

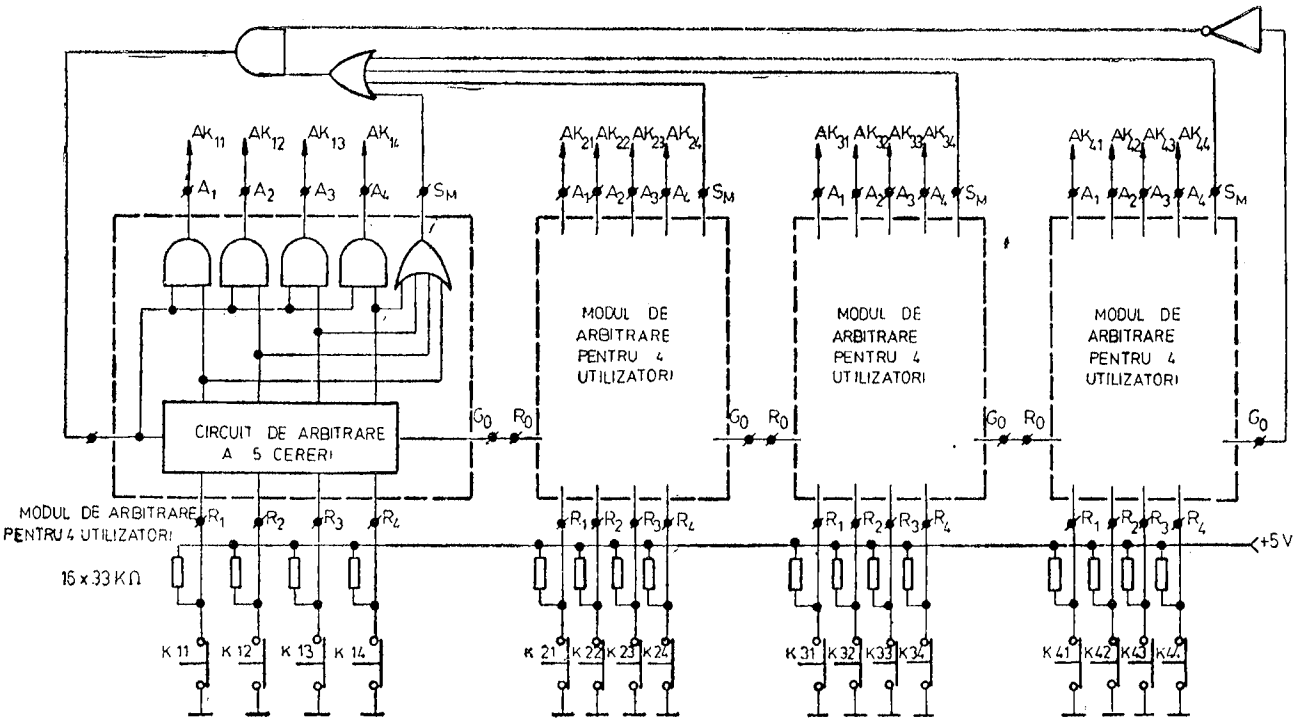


Fig. 4.4. Filtrarea oscilațiilor și arbitrarea conflictului ce apare în cazul tastărilor simultane multiple, folosind o schemă realizată pe baza circuitului din fig. 4.3

4.1.2. INTERFAȚAREA LA MICROSISTEM A UNEI TASTATURI FĂRĂ ELECTRONICĂ PROPRIE

În fig. 4.5. este prezentat modul de conectare la microsistem a unei tastaturi simple de tipul celor folosite pentru sistemele de învățare („training kit"). Tastatura constă din 24 de taste dispuse sub forma unei matrici cu 8 linii și 3 coloane. Liniile sînt conectate la „portul" *A* iar coloanele la bornele *PC*4, *PC*5 și *PC*6 ale „portului" *C* ale unei interfețe programabile 8255. Dispunerea

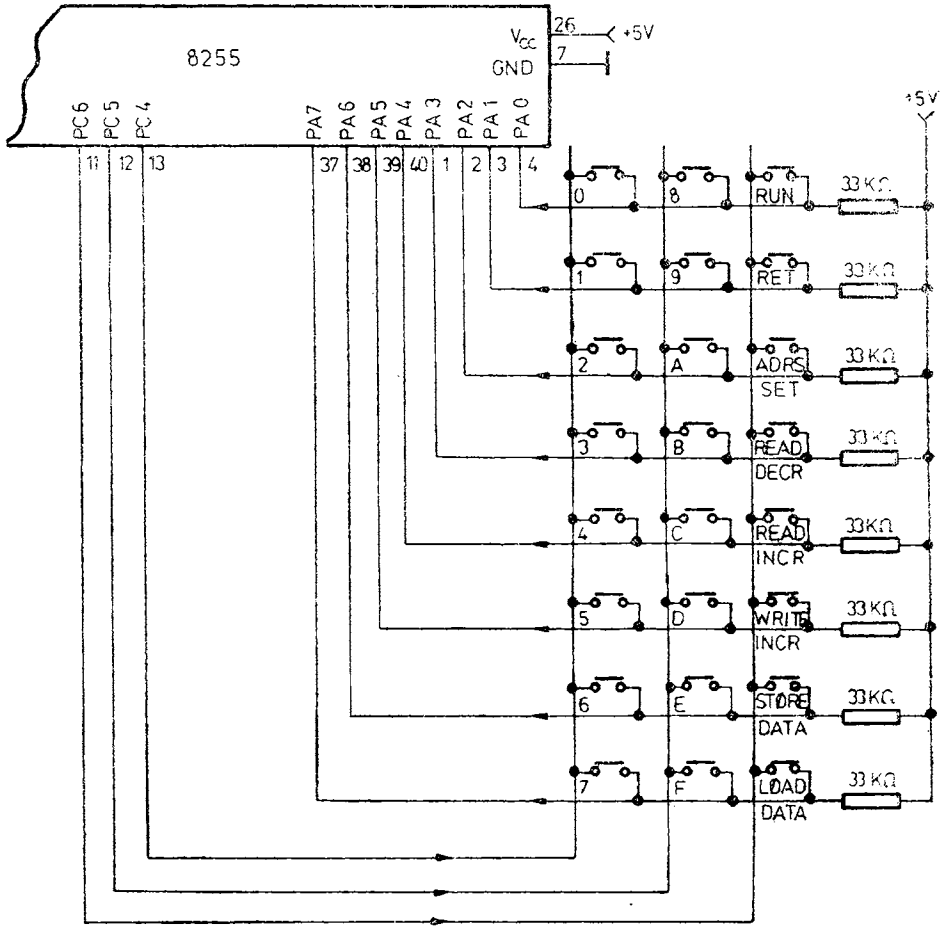


Fig. 4.5. Conectarea tastaturii la microsistem

celor 24 de taste sub forma unei matrici cu 6 linii şi 4 coloane ar fi fost mai economică din punct de vedere al numărului total de (linii + coloane) faţă de soluţia descrisă în fig. 4.5 [10 (linii + coloane) comparativ cu 11 (linii + + coloane)]. Deoarece însă este preferabilă utilizarea identică a tuturor bornelor unui „port" s-a optat totuşi pentru soluţia din fig. 4.5 care corespunde acestui deziderat („portul" *A* este dedicat în întregime liniilor matricii tastaturii).

Pentru identificarea prin program a tastei apăsate la un moment dat se pot folosi două metode. Prima, este o metodă de inspecţie secvenţială constînd în comutarea unui bit cu valoarea logică 0 pe rînd la fiecare coloană a matricii, menţinînd toate celelalte coloane la valoarea logică 1. Identificarea tastei apăsate se produce în momentul detectării valorii logice 0 la ieşirea uneia din liniile matricii. La intersecţia coloanei la care este aplicat în acel moment semnalul logic 0 şi linia la care este selectat acest semnal, rezultă coordonatele tastei apăsate.

Cea de-a doua metodă de identificare a tastei apăsate este metoda de inversare a liniilor şi coloanelor. Metoda este uşor aplicabilă în cazul în care este disponibilă o interfaţă programabilă pentru controlul tastaturii, existînd astfel posibilitatea inversării prin program a funcţiunilor de intrare/ieşire a „porturilor" la care sînt conectate liniile şi coloanele matricii cu taste. În fig. 4.6 se prezintă principiul de aplicare a acestei metode în cazul unei tastaturi matriciale cu 4×4 taste conectate la „portul" C al unei interfeţe programabile 8255. Metoda constă din două etape. În prima etapă (fig. 4.6 a) „portul" la care sînt conectate liniile tastaturii este programat pentru ieşire iar cel la care sînt conectate coloanele este programat pentru intrare. În această etapă în registrul de date al „portului" de ieşire se înscrie peste tot valoarea logică 0. La toate intrările „portului" de intrare, cu excepţia celei corespunzătoare coloanei pe care se află tasta apăsată, se menţine însă valoarea logică 1. Numai pe coloana pe care se află tasta apăsată apare un semnal avînd valoarea logică 0. Astfel în prima etapă, pe baza informaţiei ce se citeşte în „portul" de intrare, se identifică coloana pe care se află tasta apăsată (în exemplul ilustrat este apăsată tasta *K*32). În cea de-a doua etapă se inversează funcţiile de intrare/ ieşire ale celor două „porturi" ; „portul" la care sînt conectate liniile se reprogramează pentru intrare, iar cel la care sînt conectate coloanele, ca „port" de ieşire. Informaţia recepţionată în „portul" coloanelor în etapa precedentă este emisă acum pe coloanele matricei determinînd recepţionarea valorii logice 0 în bitul „portului" de intrare la care este conectată linia pe care se află tasta apăsată. În toate celelalte poziţii ale „portului" liniilor se recepţionează valoarea logică 1 corespunzînd tastelor neapăsate.

Prima metodă de identificare necesită mai multe etape pentru identificarea tastei apăsate. Procedura se poate reduce la un număr de etape egal cu minimul dintre numărul de linii şi numărul de coloane ale matricii tastaturii.

În cazul celei de-a doua metode identificarea comportă întodeauna doar două etape, indiferent de dimensiunile matricei tastaturii. Această metodă

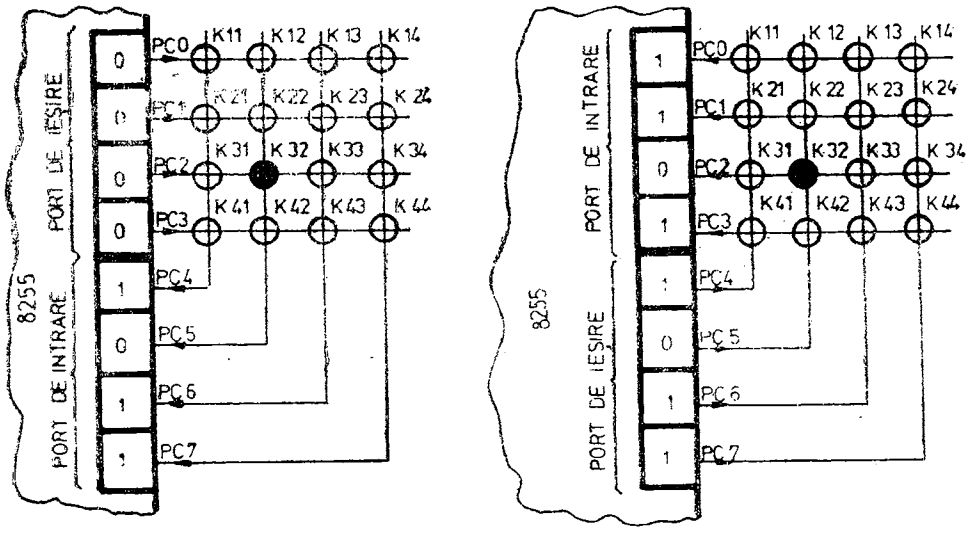


Fig. 4.6. Identificarea coordonatelor tastei apăsate prin metoda inversării liniilor şi coloanelor

este net avantajoasă în cazul tastaturilor matriciale de mari dimensiuni (de exemplu 8 ×8).

În cele ce urmează, pentru tastatura prezentată în fig. 4.5 s-a optat pentru prima metodă de identificare a tastei apăsate. S-a făcut această opţiune deoarece, pe de o parte, modul de aranjare a tastelor în matrice permite identificarea în numai trei etape iar pe de altă parte se elimină necesitatea reprogramării funcţiunilor „portului" programabil, „port" ce poate fi folosit de către monitor şi pentru alte scopuri decît controlul tastaturii. Astfel „portul" A rămîne programat ca „port" de intrare iar „portul" C ca „port" de ieşire.

4.1.3. PROGRAM DE MONITORIZARE A TASTATURII

Monitorizarea tastaturii din fig. 4.3 comportă în primul rînd identificarea tastei apăsate în condiţiile eliminării efectului oscilaţiilor contactelor mecanice. În al doilea rînd, ţinîndu-se seama de identitatea stabilită se ia decizia asupra procedurii de tratare a tastării efectuate.

Pentru identificarea tastei apăsate se inspectează tastatura coloană cu coloană prin subprogramul următor :

```
INSPTAST :  MVI   D,00H LA A,0   ; Se inițializează registrul D.
            MØV   B,D            ; Se înscrie în registrul B numărul
                                   de ordine al primei taste de pe
                                   prima coloană care se inspectează
                                   în tastatură.
            MVI   A, 92 H        ; Se încarcă în acumulator cuvîntul
                                   de control pentru „portul" progra-
                                   mabil 8255.
            ØUT   CØNTROL 8255;  Se programează registrul de control
                                   al „portului" programabil.
            MVI   A, OEFH        ; Se pregătește în acumulator cuvîn-
                                   tul ce se va încărca în „portul" C
                                   în prima etapă, de inspectare a pri-
                                   mei coloane a tastaturii (PC4=0
                                   și toate celelalte PCi=1).
            ØUT   PØRTC          ; Se încarcă în „portul" C cuvîntul
                                   pregătit în acumulator.
            IN    PØRTA          ; Se înscrie în acumulator informația,
                                   citită prin „portul" A (referitoare
                                   la starea tastelor 0, ... , 7).
            XRI   OFFH           ; Se complementează informația ci-
                                   tită în acumulator (pentru a o
                                   aduce în logică pozitivă).
            JNZ   IDENTIF        ; Dacă s-a identificat tasta apăsată
                                   pe coloana investigată se sare la
                                   adresa IDENTIF.
            MVI   B, 8           ; Se înscrie în registrul B numărul
                                   primei taste de pe coloana care se
                                   inspectează.
            MVI   A, ODFH        ; Se pregătește în acumulator conți-
                                   nutul „portului" C pentru a doua
                                   etapă a baleierii tastaturii (PC5=0
                                   și restul PCi=1).
            ØUT   PØRTC          ; Se încarcă în „portul" C cuvîntul
                                   pregătit pentru a doua etapă a
                                   baleierii tastaturii ;
            IN    PØRTA          ; Se înscrie în acumulator informația,
                                   citită prin „portul" A (referitoare
                                   la starea tastelor 8, ... , F).
            XRI   OFFH           ; Se complementează informația citită
                                   în acumulator.
```

```
                JNZ   IDENTIF     ; Dacă s-a identificat tasta apăsată
                                    pe coloana cercetată se sare la
                                    adresa IDENTIF.
                MVI,  B, 10 H     ; Se înscrie în registrul B numărul de
                                    ordine al primei taste de pe a treia
                                    coloană (care se va inspecta).
                MVI   A, OBFH     ; Se pregăteşte în acumulator conţi-
                                    nutul „portului" C pentru inspec-
                                    tarea celei de-a treia coloane de
                                    taste (PC6=0).
                ØUT   PØRTC       ; Se încarcă în „portul" C cuvîntul
                                    pregătit pentru a treia etapă a
                                    inspectării tastaturii.
                IN    PØRTA       Se înscrie în acumulator informaţia
                                    citită prin „portul" A (privind sta-
                                    rea tastelor funcţionale de pe coloana
                                    a treia).
                XRI   OFFH        ; Se complementează conţinutul acu-
                                    mulatorului.
                JNZ   IDENTIF     ; Dacă s-a identificat tasta apăsată
                                    pe coloana cercetată se sare la
                                    adresa IDENTIF.
                DCR A             Neidentificîndu-se nici o tastă apă-
                                    sată acumulatorul este anulat şi
                                    prin decrementare se face A=
                                    =OFFH.
                RET               ;
IDENTIF :       RRC               ; Se roteşte acumulatorul spre dreapta
                                    şi se încarcă în bistabilul de trans-
                                    fer conţinutul iniţial al bitului A7
                                    al acumulatorului.
                JC    IDENTIF+8   ; Dacă în bistabilul de transfer s-a
                                    găsit „1" se sare la adresa
                                    IDENTIF+8.
                INR   D           ; Se incrementează conţinutul regis-
                                    trului D reprezentînd poziţia tastei
                                    apăsate (indexul) în cadrul coloanei
                                    unde a fost identificată.
                JMP   IDENTIF     ; Se reia căutarea bitului „1" din
                                    acumulator.
IDENTIF+8 :     MØV   A, D        ; Se încarcă în acumulator indexul
                                    tastei apăsate din cadrul coloanei
                                    unde a fost identificată.
```

```
        ØRA  B          ; Se determină numărul de ordine
                          absolut al tastei apăsate (indexul+
                          +numărul de ordine al primei taste
                          din coloană).
        RET             ;
```

Subprogramul descris furnizează ca rezultat numărul de ordine absolut al tastei apăsate, sau valoarea *OFFH* dacă nu a fost identificată vreo tastă apăsată.

Pentru a se elimina efectul oscilațiilor contactelor mecanice ale tastei procedura de inspecție a tastaturii *INSPTAST* se repetă la un interval de timp mai mare decît cea mai mare durată posibilă de producere a oscilațiilor. În continuare se prezintă subrutina *TASTIN* care efectuează această inspecție repetată și care poziționează pe *OFFH* sau anulează fanionul *FANT* după cum există sau nu o tastă apăsată.

```
TASTIN:     CALL  INSPTAST      Se apelează la subprogramul de
                                inspecție a tastaturii.
            INR   A             ; Se incrementează acumulatorul al
                                cărui conținut devine nul dacă nu
                                s-a identificat nici o tastă apăsată.
            JZ    NUTASTA       ; Dacă conținutul acumulatorului
                                este nul se sare la adresa unde se
                                tratează cazul în care nu s-a iden-
                                tificat vreo tastare.
TASTIN+7:   CALL  ÎNTÎRZIERE    ; Se apelează subprogramul de gene-
                                rare a întîrzierii.
            CALL  INSPTAST      ; Se reapelează subprogramul de
                                inspecție a tastaturii.
            MØV   B, A          ; Se salvează conținutul acumula-
                                torului reprezentînd ultimul rezul-
                                tat al inspectării tastaturii.
            INR   A             ; Se incrementează acumulatorul, a
                                cărui conținut devine nul dacă nu
                                s-a identificat nici o tastă apăsată.
            JZ    NUTASTA       ; Dacă după noua inspecție a tas-
                                taturii nu s-a mai identificat apă-
                                sarea vreunei taste se sare la
                                adresa NUTASTA.
            LDA   FANT          Se încarcă în acumulator fanio-
                                nul FANT.
            ANA   A             ; Se anulează bistabilul de transfer.
            JNZ   TASTIN+7      ; Dacă fanionul FANT nu este
                                anulat se reia bucla de program
                                de la adresa TASTIN+7 din
                                care se iasă doar cînd încetează
```

```
                                    apăsarea tastei curente (fapt sesi-
                                    zat de instrucția JZ NUTASTA).
          DCR  A                ; Se decrementează acumulatorul
                                    care era nul astfel încît conți-
                                    nutul său devine A=OFFH.
TASTIN+1AH: STA  FANT           ; Fanionul tastaturii se poziționează
                                    pe OFFH sau OOH după cum s-a
                                    identificat sau nu o tastă apăsată.
          MØV  A,B              ; Se readuce în acumulator numărul
                                    de ordine absolut al tastei
                                    identificate sau valoarea OFFH
                                    după cum s-a identificat sau
                                    nu vreo tastă apăsată.
          RET                   ;
NUTASTA:  MVI  B, OFFH          ; Se introduce în B valoarea pe
                                    care ar trebui să fie în A dacă
                                    nu s-a identificat nici o tas-
                                    tare.
          JMP TASTIN+1AH        ; Se sare la adresa de unde începe
                                    poziționarea fanionului tasta-
                                    turii.
```

Întîrzierea cerută în subprogramul *TASTIN* este generată de un subprogram special

```
ÎNTÎRZIERE   : MVI   D, 48 H;
ÎNTÎRZIERE+2 : MVI   E, OCH;
ÎNTÎRZIERE+4 : DCR   E ;
               JNZ   ÎNTÎRZIERE+4 ; ;
               DCR   D            ;
               JNZ   ÎNTÎRZIERE+2 ;
               RET.
```

Subprogramul *ÎNTÎRZIERE* asigură o întîrziere de aproximativ 9.01 ms pentru un microsistem 8080 cu un cristal de cuarț de 18.432 MHz.

Pe baza subprogramului *TASTIN* (care apelează subprogramele *INSPTAST* și *ÎNTÎRZIERE*) se concepe programul principal care monitorizează tastatura.

```
MØNITAST: CALL  TASTIN  ; Se apelează subprogramul de identifi-
                            care a tastei apăsate (prin conținutul
                            lui A) și a stării tastaturii (prin
                            FANT).
          MØV   B, A    ; Se salvează conținutul acumulato-
                            rului.
```

```
LDA   FANT      ;| Se testează starea tastaturii progra-
                ;| mului rămînînd în buclă pînă cînd
ANA   A         ;| se apasă o tastă.
JZ    MONITAST  ;|
MØV   A, B      ; Se readuce în acumulator conținutul
                  său inițial (numărul de ordine absolut
                  al tastei ce este ținută apăsată).
ANI   10H       ;| Se testează dacă tasta apăsată este
                ;| cifrică sau funcțională.
JZ    TCIFRH    ;| verificînd printr-o extragere cu mască
                  dacă numărul ei de ordine absolut
                  este mai mic respectiv mai mare sau
                  egal cu 10H. Dacă e tastă cifrică se
                  sare la procedura de tratare a tastelor
                  cifrice.
MØV   A,B       ; Se reface din nou conținutul inițial al
                  acumulatorului.
ANI   OFH       ; Deoarece sînt taste funcționale se
                  extrage prin mascare indexul lor din
                  cadrul coloanei cu acest tip de taste.
MVI   B, 0      ;| În registrele (B, C) se introduce
ADD   A         ;| dublul indexului tastelor funcționale.
MØV   C, A      ;|
LXI   H, TABL   ; Se încarcă în (H, L) adresa de început
                  a tabelului cu adresele subprogra-
                  melor de tratare a tastelor funcționale.
DAD   B         ; Se indexează (H, L) cu dublul indexului
                  tastei funcționale din cadrul coloanei
                  respective din tastatură.
MØV   A, M      ;| Se încarcă în (H, L) adresa absolută
INX   H         ;| a procedurii de tastare a tastei func-
MØV   H,M       ;| ționale apăsate.
MØV   L,A       ;|
PCHL            ; Se face salt indirect la procedura de
                  tratare a tastei funcționale apăsate.
```

Tabelul cu adresele absolute ale subprogramelor de tratare a tastelor funcționale are structura :

```
TABL     : ADR„RUN"      ;
TABL+2   : ADR„RET"      ;
TABL+4   : ADR„ADRS SET" ;
TABL+6   : ADR„READ DECR" ;
TABL+8   : ADR „READ INCR" ;
```

TABL+0AH : ADR „WRITE INCR" ;
TABL+0CH : ADR „STØRE DATA" ;
TABL+0EH : ADR „LØAD DATA".

Folosind fanionul de stare FANT programul de monitorizare asigură o interpretare corectă a tastărilor şi în sensul că o apăsare oricît de lungă este considerată ca o singură tastare. Decizia de acţionare separată presupune revenirea tastaturii în stare de repaus pentru o durată mai mare de 9 ms.

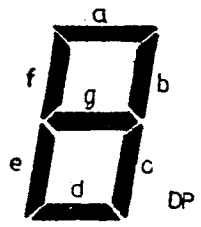
4.2. CONECTAREA UNUI DISPOZITIV DE AFIŞARE CU ELEMENTE ÎN 7 SEGMENTE [1], [2]

Dispozitivele de afişare cu elemente cu *LED*-uri, pentru sinteza cifrelor prin 7 segmente sînt folosite în construcţia majorităţii sistemelor cu microprocesoare destinate învăţării („training kit"). Aceste dispozitive permit afişarea informaţiei direct în forma hexazecimală, mai uşor de interpretat decît forma binară.

Pentru a se asigura un compromis acceptabil între cantitatea de informaţie afişată şi cost, la majoritatea sistemelor cu microprocesoare pentru învăţare se utilizează dispozitive de afişare permiţînd afişarea a 8 cifre hexazecimale.

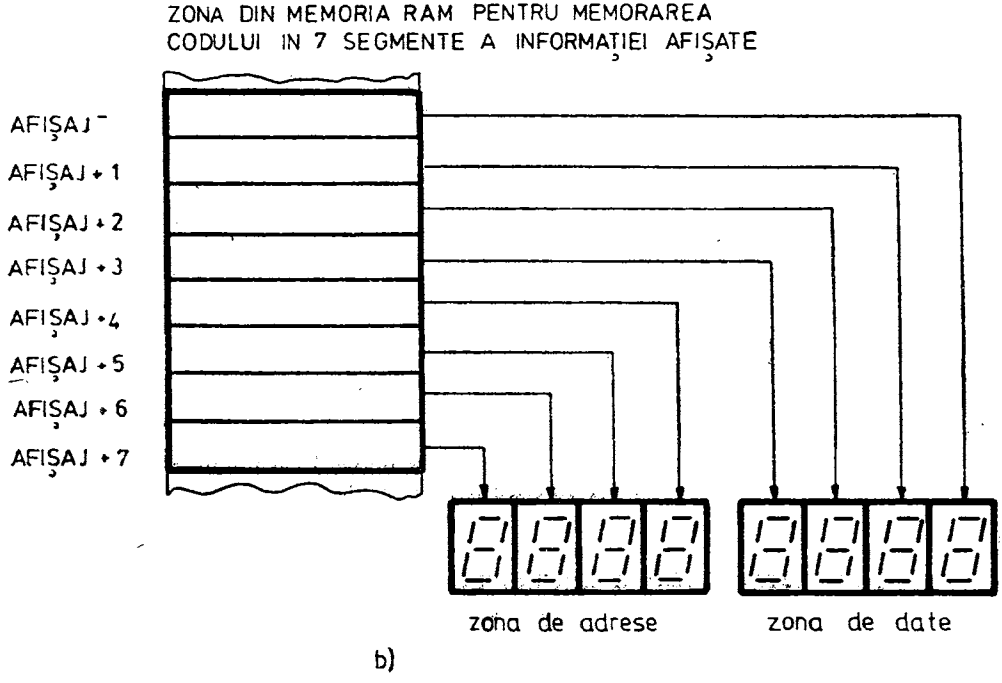
Din dorinţa reducerii costului, dispozitivele de afişare sînt de tip dinamic, cu multiplexare. Acest tip de afişaj se bazează pe inerţia optică a ochiului, informaţia fiind afişată prin baleierea ciclică a tuturor elementelor dispozitivului de afişare. Informaţia care se afişează se memorează într-o zonă specială a memoriei *RAM* a sistemului. Pentru a se elimina circuitele de conversie a codurilor, informaţia care se afişează este memorată chiar în codul de 7 segmente, pentru fiecare caracter afişat fiind necesari 8 biţi (7 biţi pentru segmente şi un bit pentru „punctul zecimal"). Conversia datelor din codul hexazecimal în codul 7 segmente este executată de un subprogram al monitorului.

În fig. 4.7. a) este reprezentată convenţia adoptată pentru corespondenţa dintre formatul în 7 segmente şi codul sub care memorează informaţia corespunzătoare unui caracter hexazecimal. În fig. 4.7.b) este reprezentată corespondenţa dintre adresele zonei de memorie rezervată pentru afişarea şi elementele dispozitivului de afişare. Corepunzător celor 8 elemente care compun dispozitivul de afişare, zona de memorie respectivă are o lungime de 8 octeţi începînd de la adresa *AFIŞAJ* inclusiv.



a : BIT 0
b : BIT 1
c . BIT 2
d : BIT 3
e BIT 4
f BIT 5
g . BIT 6
DP BIT 7

a)



b)

Fig. 4.7. Corespondențele dintre formatul în 7 segmente și biți informației memorate pentru afișarea unui caracter (a) și dintre adresele succesive ale zonei de memorie rezervată pentru afișare și elementele corespunzătoare ale dispozitivului de afișare (b)

Conform convenției stabilite în fig. 4.7.a) rezultă codarea pentru cele 16 cifre hexazecimale, așa cum este prezentată în fig. 4.8. Codarea descrisă se referă la cazul cifrelor hexazecimale fără „punct zecimal".

Caracterul afişat	Informaţia memorată	Caracterul afişat	Informaţia memorată
(0H)	5CH	(8H)	7FH
(1H)	06H	(9H)	6FH
(2H)	5BH	(AH)	77H
(3H)	4FH	(BH)	7CH
(4H)	66H	(CH)	39H
(5H)	6DH	(DH)	5EH
(6H)	7DH	(EH)	79H
(7H)	27H	(FH)	71H

Fig. 4.8. Codarea informaţiei ce se memorează corespunzător celor 16 cifre hexazecimale

4.2.1. INTERFAŢAREA MICROSISTEMULUI CU DISPOZITIVUL DE AFIŞARE

În fig. 4.9 se prezintă structura detaliată a interfeţei dintre un microsistem 8080 şi un dispozitiv de afişare cu 8 elemente în 7 segmente.

Afişajul propriu-zis constă din 8 elemente de afişare cu *LED*-uri de tipul cu anod comun. Fiecare element permite afişarea unui caracter urmat de un „punct zecimal". Deoarece afişajul este activat dinamic, catozii de acelaşi nume ai tuturor celor 8 elemente sînt legaţi între ei. Anodul fiecărui element de afişare este acţionat separat prin intermediul unui tranzistor *BC* 171 avînd în baza divizorul rezistiv format din rezistenţele de 330 Ω şi 10 kΩ.

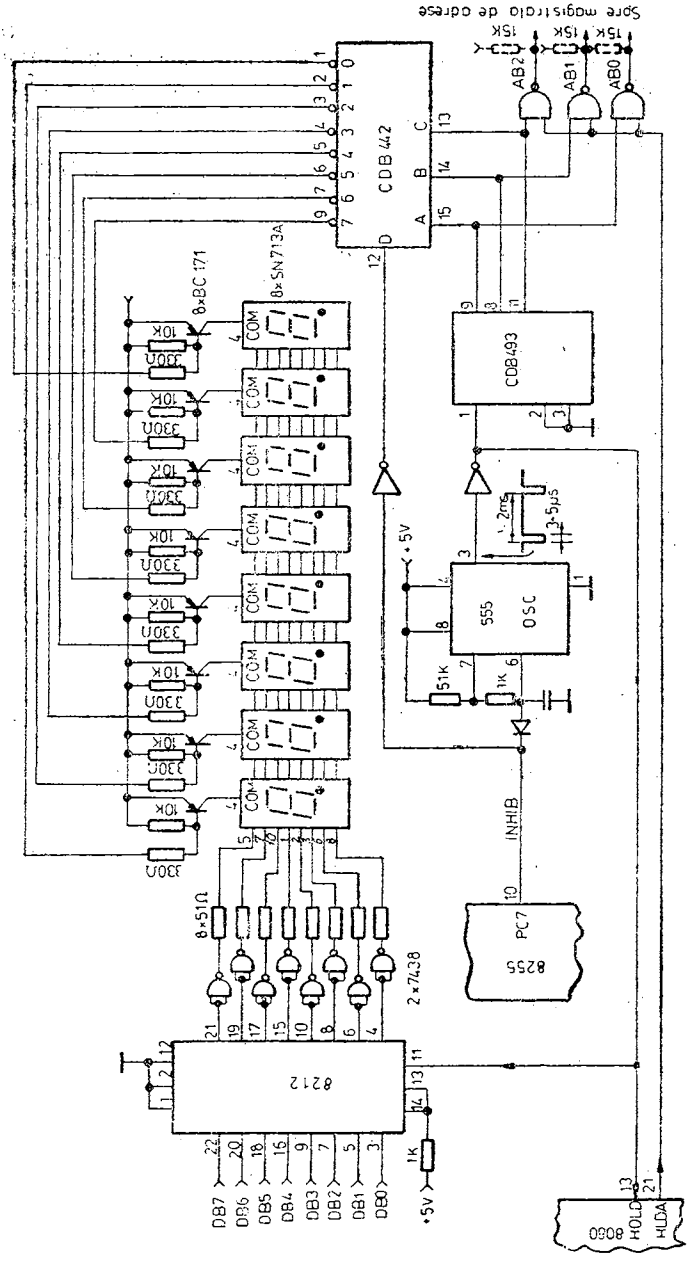


Fig. 4.9. Conectarea dispozitivului de afişare în 7 segmente la microprocesor

Codul în 7 segmente al cifrei hexazecimale ce se afișează la un moment dat este extras din memorie și este încărcat (prin intermediul magistralei de date) într-un registru tampon 8212. Ieșirile registrului tampon sînt inversate de către circuitele *ȘI—NU* de tip 7438, avînd și funcția de acționare a catozilor elementelor de afișaj cu *LED*-uri.

Baleierea anozilor este asigurată de numărătorul binar *CDB* 493 și decodificatorul binar-zecimal *CDB* 442. Folosindu-se doar trei ranguri ale numărătorului, acesta numără „modulo-8" în secvență normală, astfel încît prin decodificare se activează ciclic cîte una dintre primele 8 ieșiri ale decodificatorului. A patra intrare *D* a decodificatorului este folosită pentru validarea primelor 8 ieșiri ale acestuia. Ieșirile respective, sînt validate - ținute în starea 1 atît timp cît la intrarea *D* se aplică un semnal logic 1. Validarea acestor ieșiri este realizată prin menținerea unui semnal 0 la intrarea *D*.

La validarea decodificatorului, o singură ieșire a acestuia trece în starea 0 astfel încît tranzistorul comandat de ea se saturează, acționînd anodul elementului de afișare astfel selectat. Toate celelalte tranzistoare sînt blocate de nivelele *HI* de la ieșirile neselectate ale decodificatorului, nepermițînd astfel alimentarea anozilor celorlalte 7 elemente de afișaj. Astfel la un moment dat *LED*-urile unui singur element de afișaj sînt iluminate, corespunzător codului aflat la acel moment în registrul tampon.

Comanda numărătorului de baleiere este asigurată de către oscilatorul 555 care furnizează impulsuri negative cu durata de 3—5 μs și perioada de repetiție de 1—2 ms.

Impulsurile generate de oscilator inițiază de asemenea și procedura de *DMA* pentru extragerea codului cifrei hexazecimale care se afișează la un moment dat, de la una din adresele zonei speciale (*AFIȘAJ*, ... , *AFIȘAJ*+7) a memoriei *RAM*. Impulsurile de la oscilator, inversate pentru a fi pozitive, se aplică la intrarea *HOLD* a microprocesorului, determinînd trecerea sa în starea corespunzătoare. Semnalul de recunoaștere *HLDA* provenit de la microprocesorul aflat în starea *HOLD*, validează cele trei porți *ȘI—NU* de accees a conținutului numărătorului de baleiere la cele trei linii mai puțin semnificative, *AB0*, *AB*1 și *AB*2, ale magistralei de adrese a microsistemului. Se asigură astfel baleierea sincronă a afișajului a și zonei speciale de afișaj a memoriei *RAM*. Același impuls care declanșează procedura de *HOLD* asigură și încărcarea în portul 8212 a codului citit din *RAM* de la adresa determinată de numărătorul *CDB* 493.

Atît decodificatorul de selecție a elementelor de afișaj cît și oscilatorul care generează impulsurile de inițiere a procedurii de *DMA* pot fi inhibate prin trecerea semnalului *INHIB* în starea *LØ*. Această inhibare poate fi realizată prin program deoarece semnalul de inhibare provine de la borna *PC*7 a portului programabil 8255.

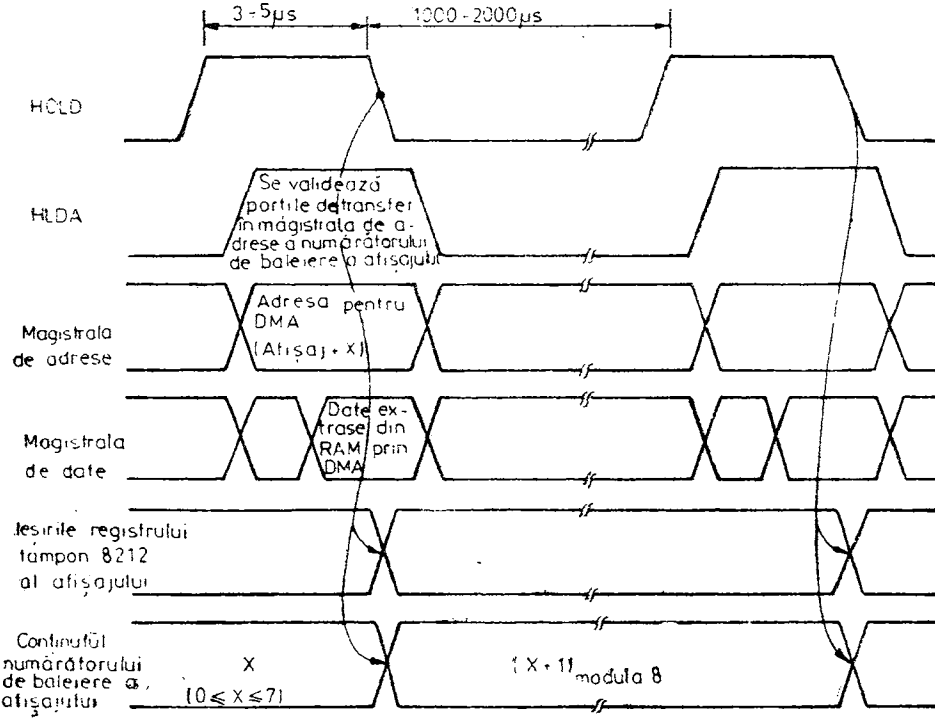


Fig. 4.10. Diagramele semnalelor corespunzătoare transferului de date din RAM în registrul tampon al afișajului prin DMA

În fig. 4.10 se prezintă diagramele de timp ale semnalelor care participă la realizarea prin *DMA* a transferului de date din *RAM* în registrul tampon 8212 al afișajului.

Se poate remarca faptul că apare un decalaj de un bit între conținutul numărătorului de baleiaj care selectează elementele de afișare și valoarea adresei de *RAM* de la care a fost extrasă informația aflată în mod curent în registrul tampon al afișajului. În schema interfeței prezentate în fig. 4.9 s-a ținut seama de acest fapt decalîndu-se în mod corespunzător conexiunile dintre ieșirile decodificatorului CDB 442 și tranzistoarele de acționare a anozilor elementelor de afișaj.

Datorită „acaparării de ciclu" care apare prin *DMA*, rularea programului pe microprocesor are loc în tranșe de cîte 1—2 ms, cu pauze de 3—5 μs. Dacă programul ce se rulează nu poate accepta astfel de pauze, procedura de *DMA* corespunzătoare afișării poate fi suspendată. Acest lucru este posibil prin progra-

marea corespunzătoare a bitului *PC7* a portului 8255, coborînd la *LØ* semnalul *INHIB* ceea ce determină blocarea oscilatorului interfeței afișajului. Evident are loc și inhibarea afișajului, elementele acestuia stingîndu-se datorită invalidării decodificatorului *CDB* 442 (fig. 4.9).

4.2.2. SUBPROGRAMUL DE CONVERSIE A DATELOR DIN HEXAZECIMAL ÎN CODUL 7 SEGMENTE

Afișarea datelor se realizează prin simpla lor plasare în zona *AFIȘAJ*... ...*AFIȘAJ*+7 a memoriei *RAM*. Înainte însă de a fi plasate în această zonă datele ce se afișează trebuie convertite din hexazecimal în codul de 7 segmente.

În fig. 4.11 se prezintă schematic operațiile care au loc în cadrul acestei conversii.

S-a considerat că cele 8 cifre hexazecimale care urmează să fie afișate se află înscrise în hexazecimal (2 cifre hexazecimale/octet) la patru adrese succesive din *RAM* începînd de la adresa *AFHEXA*.

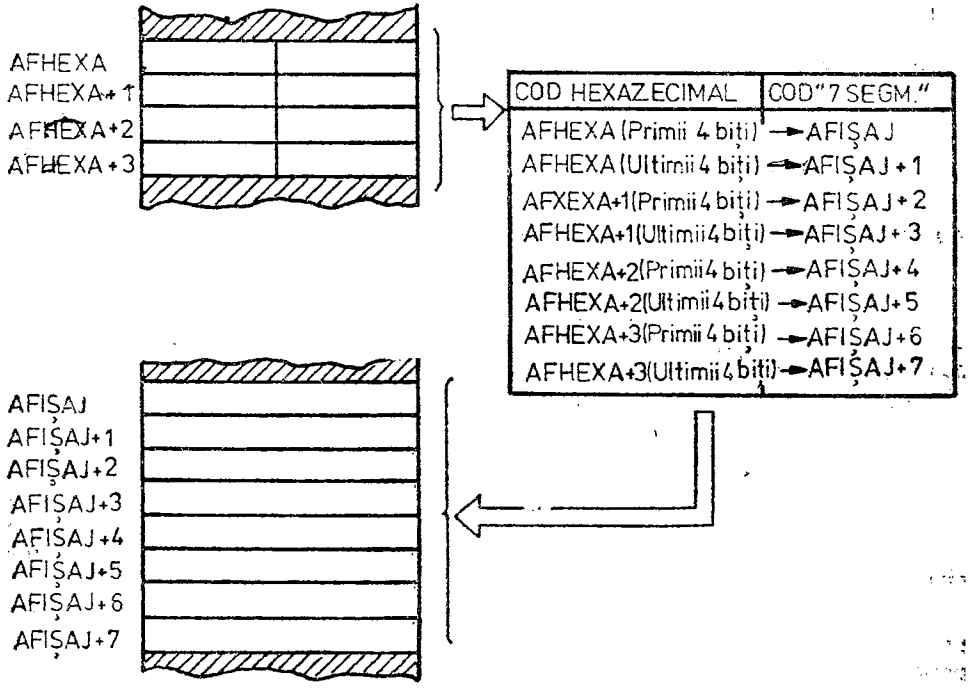


Fig. 4.11. Conversia de cod a datelor în vederea afișării

Subprogramul *CØNVH7* de conversie citește pe rînd cifrele hexazecimale din blocul *AFHEXA*, le convertește din hexazecimal în codul de 7 segmente (conform legii de conversie stabilite în fig. 4.8) și le transferă în blocul *AFIȘAJ* (conform corespondenței stabilite în fig. 4.11).

În continuare se prezintă subprogramul scris în limbaj de asamblare 8080.

```
CØNVH7    : LXI  H, AFHEXA  ; Se încarcă în (H,L) adresa de început
                              a blocului de memorie AFHEXA
                              (Se inițializează contorul blocului
                              AFHEXA).
            LXI  D, AFIȘAJ  ; Se încarcă în (D,E) adresa de început
                              a blocului de memorie AFIȘAJ
                              (Se inițializează contorul blocului
                              AFIȘAJ).
            LXI  B, SEGD    ; Se încarcă în (B,C) adresa de început
                              a tabelului cu legea de conversie a
                              codurilor.
CØNVH7+9  : MØV  A,M        ; Se încarcă în acumulator un octet
                              din blocul AFHEXA de la adresa
                              dată de contorul aflat în (H, L).
            INX  H            Se incrementează contorul de adrese
                              al blocului AFHEXA.
            PUSH H          ; Se salvează în stivă contorul de
                              adrese a blocului AFHEXA.
            PUSH PSW        ; Se salvează cuvîntul de stare pro-
                              gram (inclusiv octetul din acumu-
                              lator).
            ANI  0F0H       ; Se extrag „prin mascare" primii patru
                              biți ai acumulatorului, reprezentînd
                              cifra hexazecimală curentă.
            RRC             ;
            RRC             ; Se deplasează cu patru biți spre
            RRC             ; dreapta cifra hexazecimală extrasă
            RRC             ; „prin mascare".
            MVI  H,0        ; Se șterg registrele H și L.
            MØV  L,A        ; Se introduce în registrul L cifra
                              hexazecimală curentă.
            DAD  B          ; Adresa de început a tabelului cu
                              legea de conversie se indexează cu
                              cifra hexazecimală curentă și se intro-
                              duce în registrul de adrese (H,L)
                              al memoriei.
```

```
MØV   A,M        ; Se introduce în acumulator octetul
                   citit din tabelul de conversie, ca
                   reprezentînd codul în 7 segmente al
                   cifrei hexazecimale curente.
STAX  D          ; Codul în 7 segmente al cifrei curente
                   se memorează în blocul AFIŞAJ la
                   adresa dată de contorul din (D,E).
INX   D          ; Se incrementează contorul de adrese
                   al blocului AFIŞAJ.
PØP   PSW        ; Se restaurează cuvîntul de stare
                   program de dinaintea conversiei de
                   cod (se reface şi conţinutul acumula-
                   torului).
ANI   0FH        ; Se extrag „prin mascare" ultimii
                   patru biţi ai octetului adus din blocul
                   AFHEXA.
MVI   H, 0       ; Se şterg registrele H şi L.
MØV   L,A        ; Cei patru biţi extraşi „prin mascare"
                   se introduc în registrul L.
DAD   B          ; Adresa de început a tabelului de
                   conversie se indexează cu cifra hexa-
                   zecimală curentă şi se introduce în
                   registrul de adrese (H,L) al memoriei.
MØV   A,M        ; În acumulator se introduce codul
                   în 7 segmente al cifrei hexazecimale
                   curente.
STAX  D          ; Codul în 7 segmente al cifrei hexa-
                   zecimale curente se memorează în
                   blocul AFIŞAJ la adresa dată de
                   contorul de adrese (D,E).
PØP   H          ; Se restaurează în (H,L) contorul de
                   adrese al blocului AFHEXA.
INR   E          ; Se incrementează contorul de adrese
                   al blocului AFIŞAJ.
JNZ   CØNVH7+9   ; Se repetă ciclul pînă la completarea
                   tuturor celor 8 adrese ale blocului
                   AFIŞAJ (Plasarea blocului s-a ales
                   astfel încît AFIŞAJ+8 să aibă
                   ultimii doi octeţi egali cu 00H,
                   valoare convenabilă pentru testarea
                   condiţiei de ieşire din ciclu).
RET              ; Revenirea din subprogram.
```

Tabelul de conversie a codurilor este plasat în memorie sub forma unui bloc de 16 octeți, începînd de la adresa *SEGD*.

```
SEGD :  5CH  ;
        06H  ;
        5BH  ;
        4FH  ;
        66H  ;   Valorile hexazecimale ale octeților conținînd codul
        6DH  ;   în 7 segmente ale cifrelor hexazecimale, conform
        7DH  ;   fig. 4.2.2.
        27H  ;
        7FH  ;
        6FH  ;
        77H  ;
SEGD :  7CH  ;
        39H  ;
        5EH  ;
        79H  ;
        71H  ;
```

4.3. CONECTAREA UNUI CALCULATOR DE BIROU CU IMPRIMANTĂ LA UN MICROSISTEM

Calculatorul de birou cu imprimantă are deja implementată în el electronica de control și acționare a mecanismului imprimantei. Folosindu-se aceste funcțiuni existente se poate simplifica dialogul microprocesorului cu imprimanta. Dialogul este redus la un dialog de „tastare a claviaturii calculatorului", care necesită o interfață și programe mai simple decît în cazul conectării directe a imprimantei la microprocesor.

În continuare se prezintă modul de conectare la microprocesor a unui calculator de birou cu imprimantă *FELIX CE*-129 *T*, fabricat în țara noastră. Calculatorul poate fi astfel folosit ca periferic pentru extragerea datelor numerice din microprocesor.

4.3.1. MODUL DE UTILIZARE A CALCULATORULUI DE BIROU FELIX CE-129T

Calculatorul *FELIX CE*-129 *T* este realizat în jurul unității centrale reprezentată de circuitul integrat pe scară largă *MOS*, de tipul *HD* 35107 (fig. 4.12). Acest circuit execută toate operațiile aritmetice și de control a schimbului de informații cu tastatura și imprimanta.

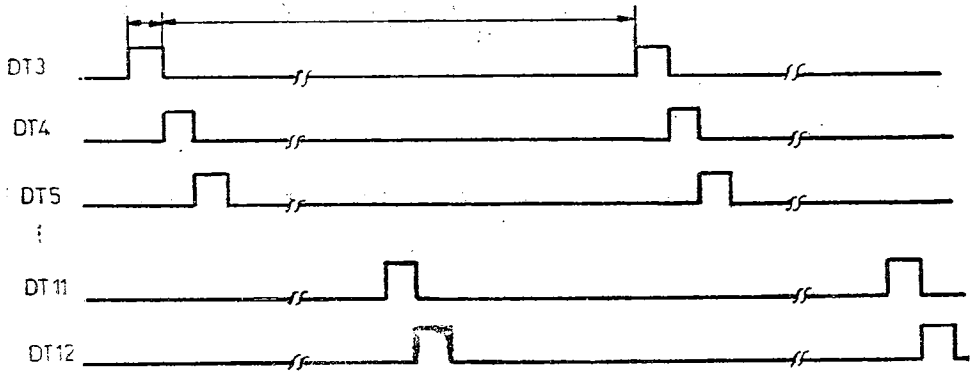


Fig. 4.12. Impulsurile de explorare a tastaturii

Pentru scopul propus interesează doar modul în care se realizează schimbul de informații a acestui circuit cu tastatura calculatorului.

Introducerea în calculator a datelor de la tastatură se realizează identificînd tasta apăsată la un moment dat printr-o procedură de multiplexare-demultiplexare temporală (scanare).

În acest scop unitatea centrală a calculatorului emite în permanență impulsurile de explorare, *DT*3, *DT*4, ... *DT*12. Aceste impulsuri (fig. 4.13) sînt decalate astfel încît la un moment dat doar unul din ele poate fi prezent.

Tastele calculatorului sînt dispuse sub formă de matrice dreptunghiulară cu 10 coloane, reprezentînd intrările, și 4 linii reprezentînd ieșirile din matrice. La intrările matricii se aplică cele 10 secvențe de impulsuri de explorare *DT*3, *DT*4, ..., *DT*12. La dispunerea tastelor pe liniile matrici s-a avut în vedere gruparea pe fiecare linie a tastelor cu același rol funcțional (fig. 4.12).

Tastele de comandă a operațiilor aritmetice și de acționare a imprimantei au fost grupate pe primele două linii, corespunzîndu-le ieșirile *LKS* și *FKS*. Cele zece taste pentru cifrele zecimale au fost grupate pe linia a treia, avînd ca ieșire semnalul *NKS*. Pe linia a patra avînd ca ieșire semnalul *SDP* au fost dispuse contactele cursorului de selecție a poziției virgulei.

Atunci cînd se apasă o tastă a calculatorului se închide de fapt contactul normal deschis dintre coloana și linia corespunzătoare nodului de matrice în care se află tasta. Ca urmare la ieșirea respectivă apare secvența de impulsuri *DTi i*, $\in\{3, 4 \ldots, 12\}$, corespunzătoare coloanei pe care se află tasta apăsată.

La ieșirile *LKS*, *FKS* și *NKS* apare deci un semnal doar pe durata apăsării vreuneia dintre tastele calculatorului. La ieșirea *SDP* a matricii apare în permanență secvența de impulsuri *DTi* corespunzătoare poziției cursorului de selecție a virgulei.

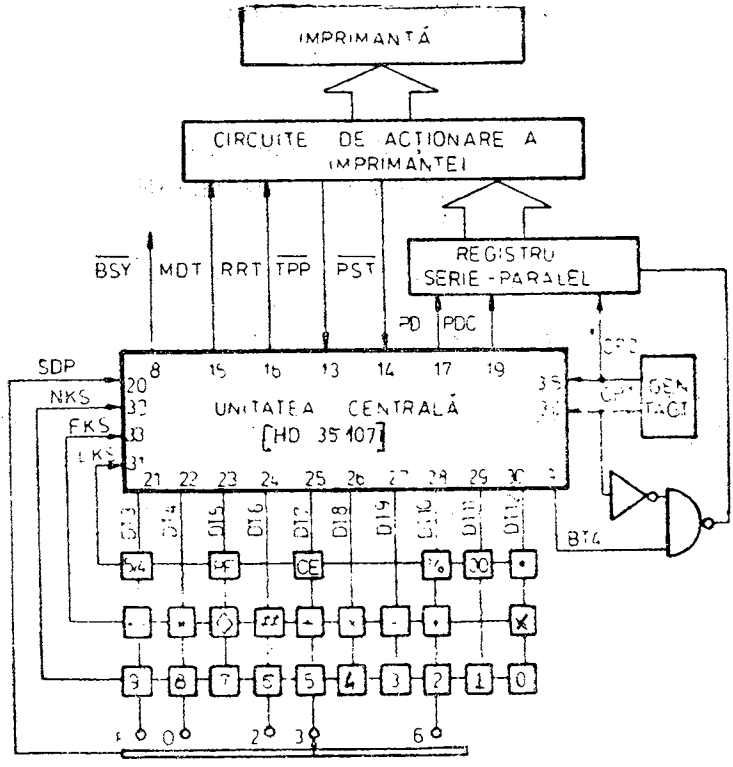


Fig. 4.13. Structura calculatorului de birou FELIX CE 129-T

Tasta apăsată la un moment dat este recunoscută de către unitatea centrală a calculatorului după ieşirea de matrice la care apare un semnal la un moment dat şi după tipul de impulsuri care apar la acea ieşire. Există, impuse de către fabricant, anumite durate minime de apăsare a tastelor şi pentru pauza între două tastări succesive.

Unitatea centrală a calculatorului de birou controlează şi întregul proces de tipărire a datelor la imprimantă. Datele ce urmează a fi tipărite sînt transmise în serie spre registrul serie-paralel în care se asamblează linia ce urmează să fie tipărită. Prin semnalul *MDT* unitatea centrală comandă motorul de acționare al imprimantei iar prin semnalul *RRT* comandă culoarea panglicii tușate cu care se face tipărirea (numerele pozitive fiind tipărite cu negru iar cele negative cu roşu).

Sincronizarea imprimării este asigurată de către două semnale trimise de imprimantă spre unitatea centrală : semnalul $\overline{TPP}$ care marchează poziția discurilor de tipărire și $\overline{PST}$ care marchează terminarea tipăririi.

4.3.2. INTERFAȚA DINTRE MICROPROCESORUL 8080 ȘI CALCULATORUL DE BIROU CU IMPRIMANTA FELIX CE-129T

Interfațarea se realizează la nivelul tastaturii calculatorului de birou. Practic se dublează tastatura mecanică existentă cu o schemă electronică de multiplexare compatibilă din punct de vedere funcțional și electric cu unitatea centrală a calculatorului de birou.

În fig. 4.14 se prezintă structura interfeței.

Microprocesorul prezintă datele (codul binar al cifrelor și al ordinelor pentru imprimantă) la ieșirile unui circuit 8212 („port" de ieșire). Bitul cel mai semnificativ, *D*7, este utilizat pentru validarea ieșirilor interfeței spre calculator. Ceilalți biți : *D*6, *D*5, ..., *D*0 sînt utilizați pentru codarea datelor.

Deoarece schemele de selecție sînt realizate cu circuite integrate *TTL*, este necesar ca ieșirile acestor scheme să sufere o translatare de nivel *TTL/MOS* pentru a se asigura compatibilitatea electrică cu calculatorul de birou.

La rîndul lor impulsurile de selecție *DT*3, ..., *DT*12 furnizate de unitatea centrală a calculatorului sînt întîi translatate *MOS/TTL* și abia apoi sînt aplicate la schemele de selecție ale interfeței. Aceeași transformare se aplică și semnalului $\overline{BSY}$ care indică timpul de ocupare a unității centrale a calculatorului de către operația în curs de execuție.

Sincronizarea transferului de date dinspre microprocesor spre calculatorul de birou se realizează prin menținerea microprocesorului în starea *WAIT* pe toată durata cerută de calculatorul de birou pentru prelucrarea informației din „portul" de ieșire. Această sincronizare este asigurată de cele două monostabile : *TEMP* și *FORM* care împreună comandă bistabilul a cărui ieșire se aplică la intrarea *RDYIN* a circuitului 8224 de generare a semnalelor de orologiu.

Anularea „portului" de ieșire, în vederea asigurării pauzei corespunzătoare intervalului minim cerut de calculator între două tastări succesive, este asigurată de către partea de sincronizare a interfeței.

Pentru codarea datelor ce se introduc în „portul" de ieșire în vederea imprimării s-a folosit formatul indicat în tabelul 4.1. De asemenea în tabel sînt specificate și impulsurile de explorare *DTi* care trebuie să apară la ieșirile interfeței în funcție de informația codată, prezentată la un moment dat în „portul" de ieșire.

În cazul descris în tabel nu s-au folosit toate funcțiunile posibile ale calculatorului ci doar cele necesare pentru tipărirea datelor. A fost astfel posibilă

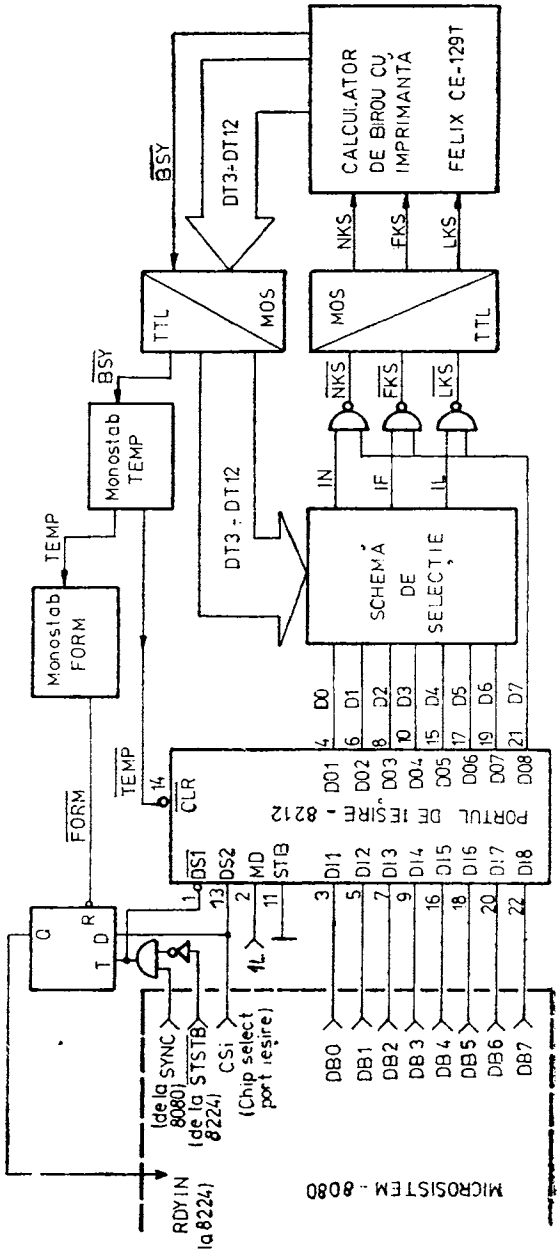


Fig. 4.14. Interfața dintre microsistemul 8080 și calculatorul de birou

Tabelul 4.1.

Informația din portul de ieșire								Tasta ce corespunde la calculator	Ieșirile interfeței		
*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0		*NKS*	*FKS*	*LKS*
1	0	0	0	0	0	0	0	0	*DT*12	0	0
1	0	0	0	0	0	0	1	1	*DT*11	0	0
1	0	0	0	0	0	1	0	2	*DT*10	0	0
1	0	0	0	0	0	1	1	3	*DT*9	0	0
1	0	0	0	0	1	0	0	4	*DT*8	0	0
1	0	0	0	0	1	0	1	5	*DT*7	0	0
1	0	0	0	0	1	1	0	6	*DT*6	0	0
1	0	0	0	0	1	1	1	7	*DT*5	0	0
1	0	0	0	1	0	0	0	8	*DT*4	0	0
1	0	0	0	1	0	0	1	9	*DT*3	0	0
1	0	0	1	1	1	1	1	*PF* (avans hîrtie)	0	0	*DT*5
1	0	1	1	1	1	1	1	. (punct zecimal)	0 0	0 0	*DT*12
1	1	0	1	1	1	1	1	$+ \rightleftarrows -$ (schimbare semn)	0	*DT*3	0
1	1	1	1	1	1	1	1	⋕ (tipărire)	0	*DT*6	0

o codare mai convenabilă a datelor din „portul" de ieșire. Biții *D*6, *D*5, *D*4 sînt folosiți pentru codarea comenzilor de imprimare. Biții *D*3, *D*2, *D*1, *D*0 sînt folosiți pentru codarea cifrelor zecimale.

Din tabel rezultă imediat ecuațiile logice pentru sinteza schemelor de selecție :

$$NKS=(CZ0.DT12+CZ1.DT11+CZ2.DT10+CZ3.DT9+CZ4.DT8+$$
$$+CZ5.DT7+CZ6.DT6+CZ7.DT5+CZ8.DT4+CZ9.DT3).D7\ ; \qquad (4.1)$$

unde CZi ; i=0, 1, ... 9 reprezintă cifrele zecimale definite de ecuaţiile binare :

$$CZ0=\overline{D3}.\overline{D2}.\overline{D1}.\overline{D0} \quad ; \tag{4.2}$$

$$CZ1=\overline{D3}.\overline{D2}.\overline{D1}.D0 \quad ; \tag{4.3}$$

$$CZ2=\overline{D3}.\overline{D2}.D1.\overline{D0} \quad ; \tag{4.4}$$

$$CZ3=\overline{D3}.\overline{D2}.D1.D0 \quad ; \tag{4.5}$$

$$CZ4=\overline{D3}.D2.\overline{D1}.\overline{D0} \quad ; \tag{4.6}$$

$$CZ5=\overline{D3}.D2.\overline{D1}.D0 \quad ; \tag{4.7}$$

$$CZ6=\overline{D3}.D2.D1.\overline{D0} \quad ; \tag{4.8}$$

$$CZ7=\overline{D3}.D2.D1.D0 \quad ; \tag{4.9}$$

$$CZ8=D3.\overline{D2}.\overline{D1}.\overline{D0} \quad ; \tag{4.10}$$

$$CZ9=D3.\overline{D2}.\overline{D1}.D0 \quad ; \tag{4.11}$$

Pentru celelalte două ieşiri rezultă ecuaţiile logice :

$$LKS=D7.\overline{D6}.\overline{D5}.D4.DT5+D7.\overline{D6}.D5.D4.DT12 ;$$

$$FKS=D7.D6.\overline{D5}.D4.DT3+D7.D6.D5.D4.DT6 ;$$

Pe baza tuturor acestor ecuaţii s-au sintetizat cu circuite integrate TTL uzuale schema de selecţie pentru cifre şi pentru funcţii aşa cum sînt reprezentate în fig. 4.15.

Pentru decodificarea codurilor binare ale cifrelor zecimale s-a folosit un circuit integrat *CDB* 442.

Al doilea decodificator binar-zecimal *CDB* 442 serveşte la decodificarea comenzilor. Fiind necesari doar trei biţi pentru codarea comenzilor, în cazul considerat a fost posibilă utilizarea celei de a patra intrări a decodificatorului ca intrare de validare pentru bitul *D7*.

Folosirea unor porţi *ŞI-NU* cu colectorul în gol, a căror ieşiri pot fi conectate prin *SAU*-cablat a făcut posibilă simplificarea schemelor de multiplexare.

Circuitele de translatare *TTL/MOS* constau fiecare dintr-o rezistenţă de 1*K* şi un tranzistor *p—n—p* (*BC* 251). Rezistenţa serveşte şi ca rezistenţă de ieşire pentru porţile *ŞI-NU* cu colectorul în gol folosite în ultimul etaj al schemelor de selecţie. Circuitele de translatare folosesc ca rezistenţă de colector rezistenţele de polarizare a bornelor respective din calculatorul de birou.

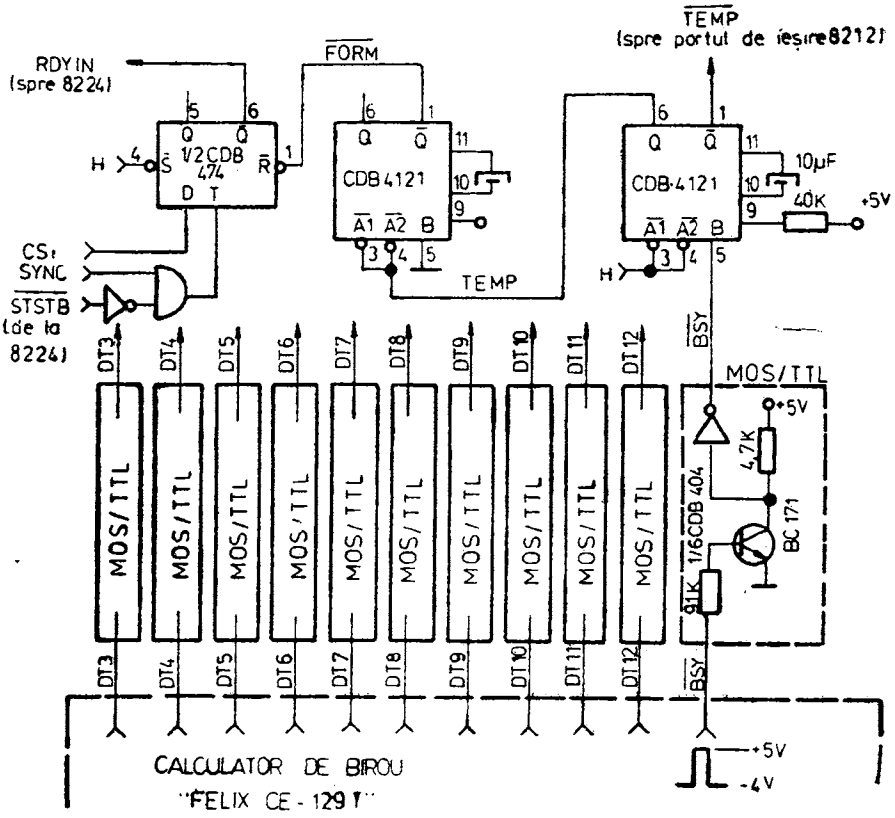


Fig. 4.16. Circuitele de translatare MOS/TTL şi de schema logică de sincronizare

În fig. 4.16 se prezintă schema detaliată a circuitelor de translatarea de nivel *MOS/TTL* şi a părţii de sincronizare a interfeţei.

Un circuit de translatare *MOS/TTL*: constă dintr-un inversor realizat pe baza unui tranzistor npn (*BC* 171) care asigură translatare de nivel propriu-zisă. Al doilea inversor, integrat, este necesar din considerente de logică, pentru a nu se complementa semnalul de intrare.

În interfaţa realizată s-au folosit 11 astfel de circuite identice pentru translatarea de nivel *MOS/TTL*: zece pentru impulsurile defazate *DTi* şi unul pentru semnalul $\overline{BSY}$ de marcare a timpului în care calculatorul este ocupat.

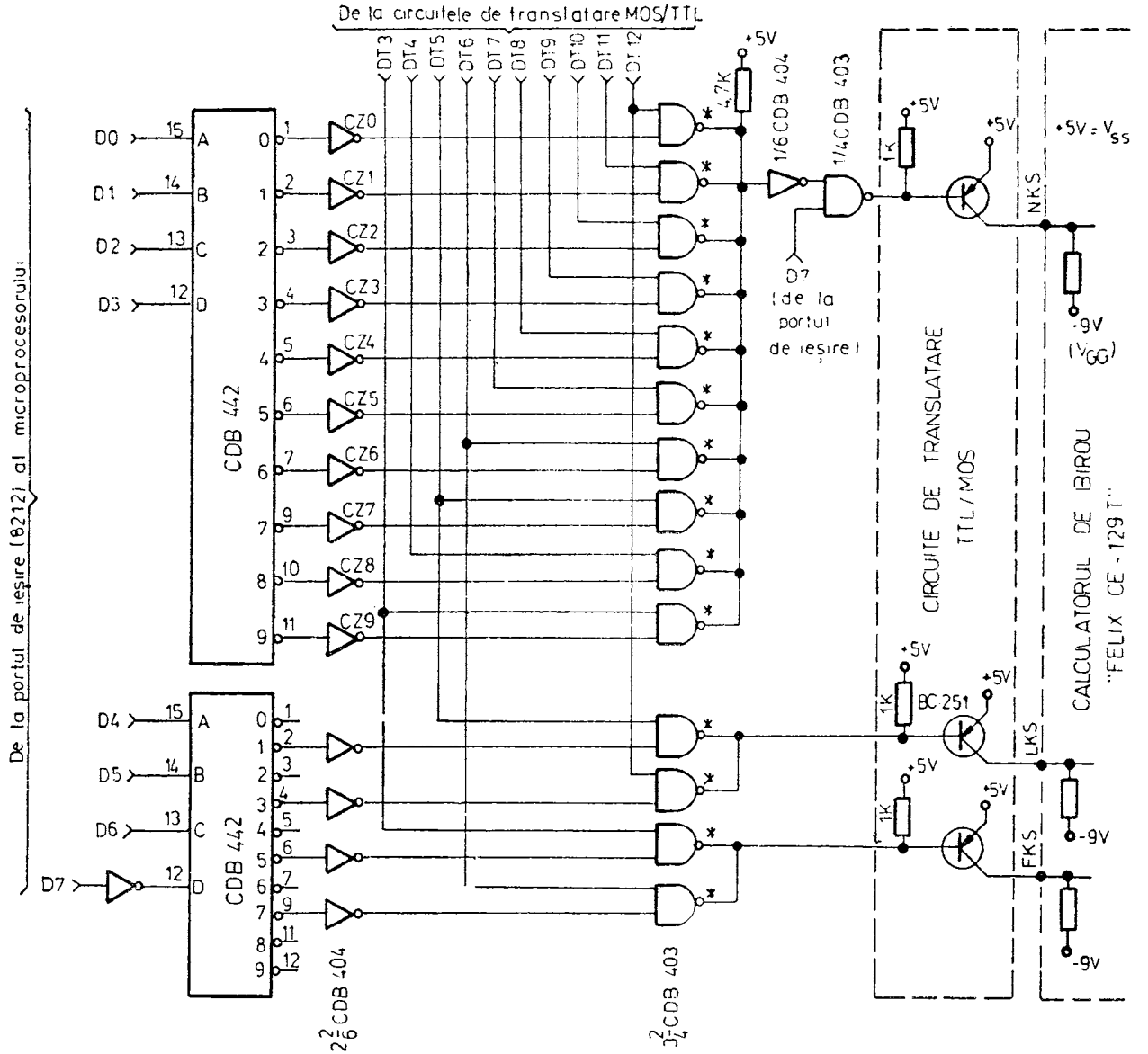


Fig. 4.15. Schema logică pentru selecția cifrelor și a funcțiilor

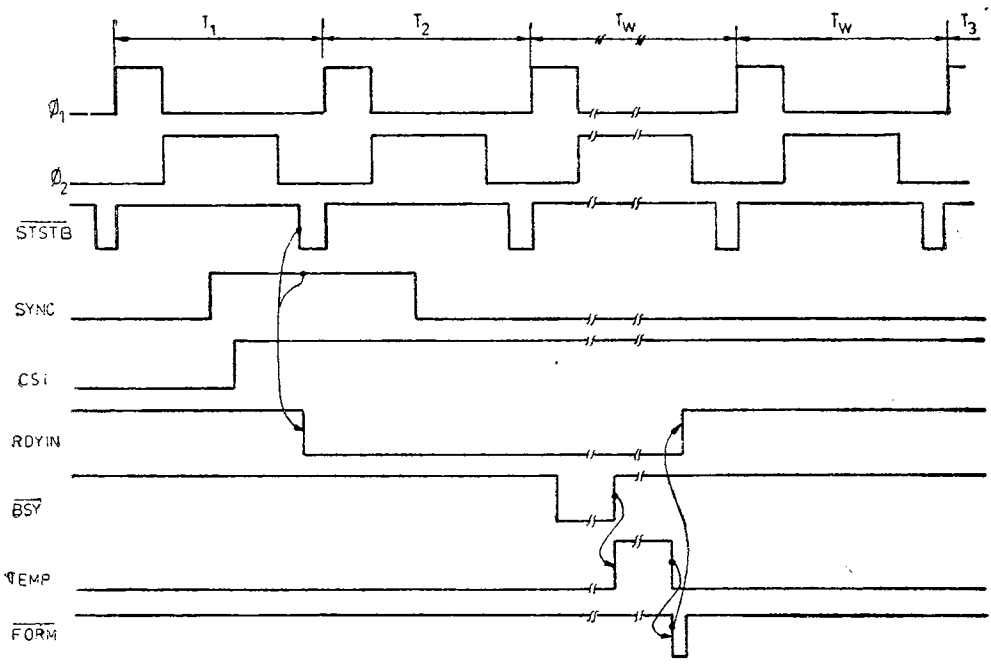


Fig. 4.17. Diagramele de timp corespunzătoare schemei de sincronizare

Funcționarea schemei de sincronizare este ilustrată de diagramele de timp din fig. 4.17.

Este posibil ca semnalul de selecție *CSi* a „portului" de ieșire să apară înaintea semnalului *SYNC*. Pentru a se asigura înscrierea corectă a informațiilor în „portul" de ieșire și în bistabilul *RDYIN* s-a folosit ca semnal de tact semnalul compus : *SYNC-STSTB*. Acest semnal de tact apare o singură dată, în starea *T*1 a ciclului de mașină (datorită lui *SYNC*). Apariția semnalului de tact se produce după stabilirea adreselor, datorită condiționării sale de *STSTB* care apare după terminarea lui Ø2 a stării T_1 și înainte de Ø1 a stării T_2.

Deci la sfîrșitul stării T_1 a instrucției de extragere vizînd calculatorul cu imprimantă, semnalul *RDYIN* ia valoarea 0. Acest semnal se aplică circuitului 8224-generatorul de semnal de orologiu a microprocesorului — determinînd trecerea microprocesorului din starea T_2 în starea T_w de așteptare.

Microprocesorul rămîne în așteptare pînă cînd unitatea centrală a calculatorului cu imprimantă nu semnalizează, prin ridicarea la 1 a semnalului $\overline{BSY}$, că a terminat de prelucrat datele primite prin „portul" de ieșire. Revenirea lui $\overline{BSY}$ înapoi la valoarea 1 determină declanșarea monostabilului care

furnizează semnalul *TEML* avînd o durată egală cu pauza minimă cerută între două tastări succesive.

Semnalul de la ieşirea negată $\overline{TEMP}$ se aplică la intrarea de anulare a portului 8212, forţîndu-l pe acesta în starea 0 pe durata pauzei între tastări. De abia după terminarea acestei temporizări se poate scrie o nouă informaţie în port.

Frontul posterior a semnalului *TEMP* declanşează la rîndul său un alt monostabil, de formare a unui impuls $\overline{FORM}$ necesar pentru iniţializarea bistabilului *RDYIN*.

Prin revenirea lui *RDYIN* la valoarea 1 se permite ieşirea microprocesorului din starea de aşteptare şi continuarea programului.

Pentru tipărirea datelor numerice la imprimanta calculatorului de birou, microprocesorul trebuie să depună printr-o instrucţie de extragere în portul de ieşire codul cifrei zecimale sau al ordinului care doreşte, codarea făcîndu-se conform convenţiei tabelate. Este necesară o singură instrucţie de extragere pentru fiecare cifră zecimală sau ordin de tipărire.

Dacă se utilizează şi funcţiile aritmetice ale calculatorului (prevăzîndu-se aceasta în schema de selecţie din interfaţă) devine posibilă utilizarea acestuia ca procesor aritmetic, aservit microprocesorului.

4.4. COMANDA UNUI MOTOR PAS CU PAS

4.4.1. INTRODUCERE

Motoarele electrice pas cu pas sînt printre cele mai larg folosite dispozitive pentru conversia informaţiilor numerice în lucru mecanic util. Comandate direct cu trenuri de impulsuri ele constituie un mod economic şi simplu de poziţionare numerică precisă. Prin înlocuirea servomotoarelor de c.c. funcţionînd în buclă închisă (fig. 4.18.a) cu motoare pas cu pas (*MPP*) în buclă deschisă (fig. 4.18.b) se elimină necesitatea unor convertoare *N/A*, amplificatoare de putere cu coeficient ridicat la liniaritate, traductoare de poziţie şi viteză, reţele de compensare a buclei.

Evoluţia motorului pas cu pas, restrîns iniţial la aplicaţii care solicitau performanţe mici sau medii, permite actualmente realizarea de servosisteme pentru deplasarea benzii perforate, disc flexibil, trasatoare *XY*, echipamente de desen automat, cititoare de cartele, deruloare de bandă magnetică, diverse instrumente optice şi maşini-unelte.

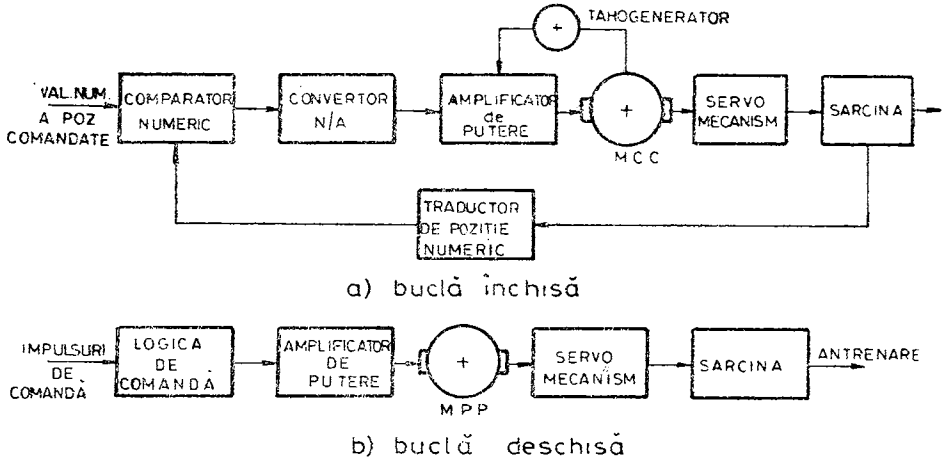


Fig. 4.18. (a, b) Realizarea servosistemelor de poziționare: a) în buclă închisă cu motor de c.c.; b) în buclă deschisă cu motor pas cu pas

4.4.2. PRINCIPIUL DE FUNCȚIONARE

Motorul pas cu pas este un motor de curent continuu cu comandă în impuls de deplasare unghiulară a rotorului proporțională cu numărul impulsurilor de comandă primite.

La fiecare impuls de comandă rotorul execută un pas unghiular α_p apoi se oprește pînă la sosirea următorului impuls. După un număr de impulsuri deplasările unghiulare se cumulează adică motorul apare ca un sumator ceea ce îl recomandă pentru acționarea de tip incremental. Fiind capabil de reversarea sensului de mișcare *MPP* poate efectua operații de însumare algebrică. Dacă este comandat corect ajunge și rămîne în sincronism cu impulsurile de comandă. De asemenea oprirea poate avea loc fără depășire inerțială eventual prin adăugarea unor dispozitive de amortizare mecanică, hidraulică sau pneumatică.

MPP permite deci deplasarea unui mecanism într-o poziție anumită, cu viteză și direcție dorită în funcție de numărul de impulsuri, frecvență și sens de rotație.

Deoarece fiecare pas rotește axul într-o poziție cunoscută singura eroare de poziționare indiferent de distanță sau direcție de mișcare depinde numai de precizia unui pas unghiular. Acesta este în mod obișnuit de 5% din valoarea unghiului α_p.

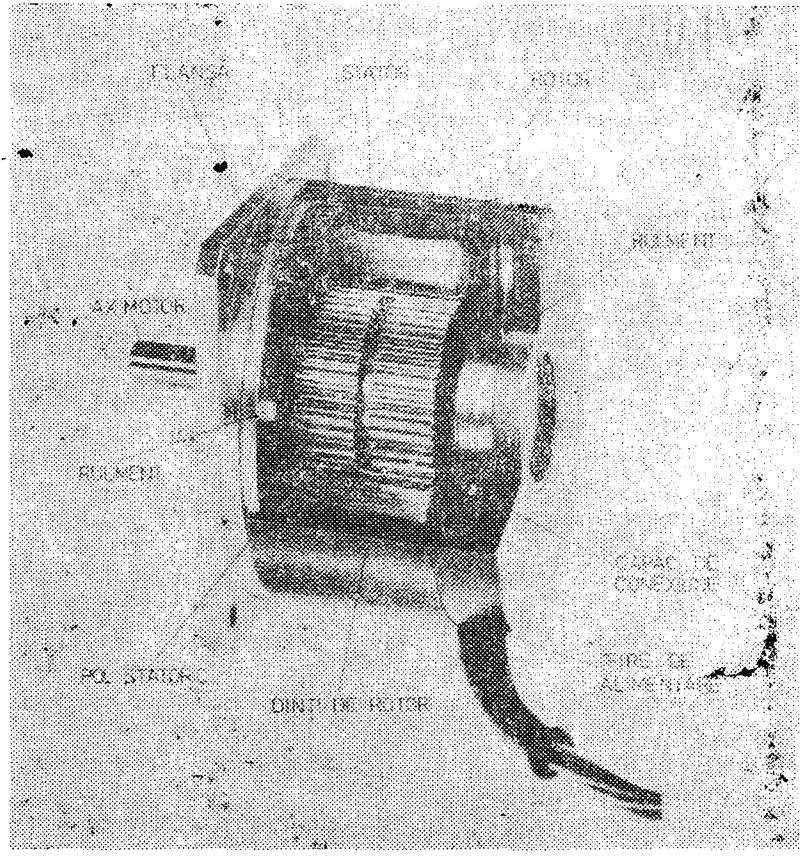


Fig. 4.19. Motor pas cu pas secționat

Numărul de paşi pentru o rotaţie completă diferă de la motor la motor. Se întîlnesc curent valori de 200, 180, 144,72, 24 şi 12 paşi pe revoluţie, corespunzînd la $1{,}8^0$, $2{,}0^0$, $2{,}5^0$, 5^0, 15^0 şi respectiv 30^0. Din punnct de vedere constructiv *MPP* pot fi clasificate în active sau reactive după tipul rotorului. La *MPP* active rotorul este realizat ca magnet permanent. Acest motor posedă cuplu ridicat, unghi de pas mare şi viteze mici de ordinul 300 paşi/s. *MPP* reactive (sau cu reluctanţă variabilă) au cupluri mai scăzute, unghi de pas mic şi viteze ridicate de pînă la 20 000 paşi/s [4].

Părţile componente ale unui *MPP* activ sînt arătate în fig. 4.19 în care se observă că rotorul are la extremităţi crestături asemănătoare unei coroane de roată dinţată ce formează polii rotorici nord la un capăt şi sud la celălalt.

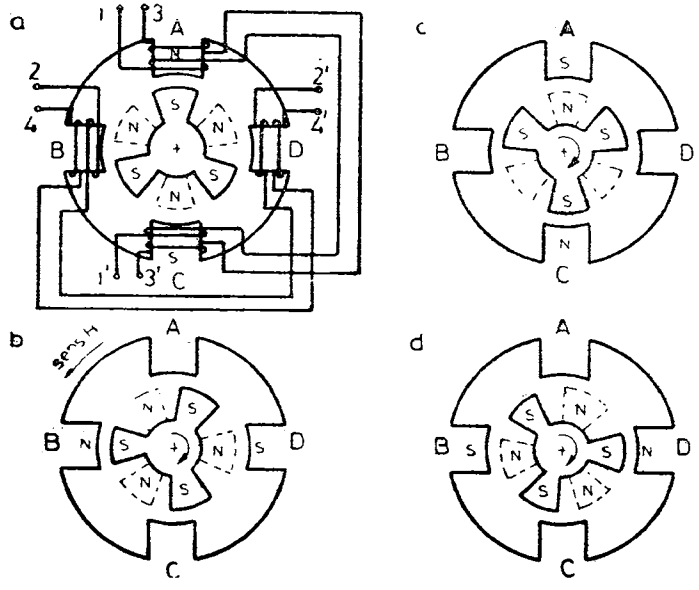


Fig. 4.20. (a, b, c, d). Principiul de funcţionare al MPP

Aceşti poli sînt reprezentaţi prin linie continuă sau întreruptă în fig. 4.20 care înfăţişează principiul de funcţionare al MPP activ. Polii statorici posedă fiecare cîte două înfăşurări conectate pentru a forma în total patru faze distincte notate $\emptyset_1 \div \emptyset_4$.

Principiul de funcţionare se bazează pe proprietăţile polilor magnetici : polii diferiţi se atrag, cei identici se resping. Deoarece numărul polilor rotorici diferă de numărul polilor statorici nu va exista niciodată o aliniere precisă între polii rotorului sau statorului. Acest fapt împreună cu secvenţa de comutare a curenţilor de excitare a înfăşurărilor determină în mod exact viitoarea mişcare a rotorului. Chiar fără alimentarea înfăşurărilor, *MPP* activ îşi păstrează poziţia deşi cu un cuplu scăzut numit cuplu rezidual.

Un exemplu de comutare a curenţilor de excitare a înfăşurărilor este arătat în fig. 4.20. În fig. 4.20. a se excită înfăşurarea 1 şi prin urmare polul *A* devine *N* iar cel opus, *C*, devine *S*. Celelalte înfăşurări nu sînt alimentate şi deci rotorul ocupă poziţia arătată. Dacă înfăşurarea 1 pierde alimentarea în schimb se alimentează înfăşurarea 2 polii *B* şi *C* prezintă un *N* respectiv un *S*. În consecinţă rotorul se aliniază aşa cum se vede în fig. 4.20.b. Pentru încă un pas se alimentează înfăşurarea 3 (fig. 4.20.c) polii *A* şi *C* devenind *S* şi *N* de această dată. După alimentarea înfăşurării 4 caz în care rotorul se aliniază conform fig. 4.20.d ciclul de comutare se repetă. Rezultă că secvenţa de excitare a înfă-

SENS ORAR				SENS ANTIORAR			
ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_1	ϕ_2	ϕ_3	ϕ_4
1	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1
0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0

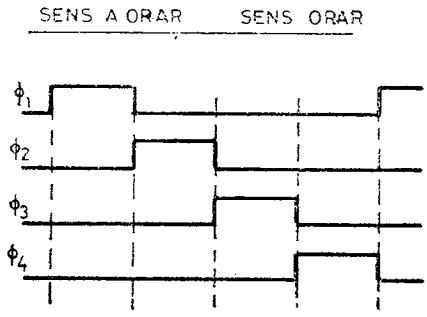


Fig. 4.21. Regimul de comandă al MPP cu jumătate de putere

şurărilor se face în ordinea 1—2—3—4—1— ... situaţie în care cîmpul magnetic creat de stator se roteşte spre stînga (antiorar) iar rotorul în sens orar. Schimbarea sensului de rotaţie înseamnă o secvenţă de comutaţie parcursă în ordine inversă, 1—4—3—2—1—1— ...

Regimul de comandă al unui *MPP* cu patru înfăşurări în care la un moment dat se excită o singură înfăşurare se numeşte regim de putere pe jumătate (tabelul şi diagrama de timp din fig. 4.21) [2].

Dacă se alimentează deodată două înfăşurări adiacente (exceptînd combinaţiile 1 cu 3 şi 2 cu 4), de exemplu 1 şi 2, apar forţe identice care produc momente a căror sumă vectorială dau o rezultantă ce aduce rotorul la jumătate de pas unghiular de alinierile cu polii *AC* sau *BD*. Dacă se întrerupe curentul de excitare prin înfăşurarea 1 şi se menţine prin înfăşurarea 2 la care se adaugă înfăşurarea 3 rotorul execută o rotire cu un pas într-o poziţie similară celei descrise înainte. Secvenţa de comandă poate continua astfel 12—23—34—41— ... iar regimul se va numi de putere întreagă (tabelul şi diagrama de timp din fig. 4.22). Bineînţeles că schimbarea sensului de rotire înseamnă parcurgerea secvenţei în ordine inversă adică 12—41—34—23—12—41— ...

SENS ORAR				SENS ANTIORAR			
ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_1	ϕ_2	ϕ_3	ϕ_4
1	0	0	1	1	0	0	1
1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	0
0	0	1	1	1	1	0	0

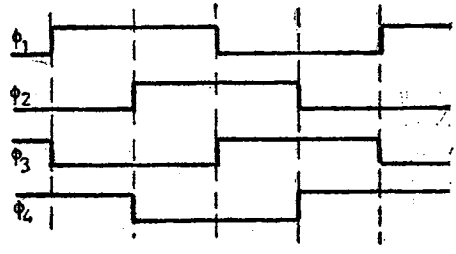


Fig. 4.22. Regimul de comandă al MPP cu putere întreagă

SENS ORAR				SENS ANTIORAR			
ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_1	ϕ_2	ϕ_3	ϕ_4
1	0	0	1	1	0	0	1
1	0	0	0	0	0	0	1
1	1	0	0	0	0	1	1
0	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0

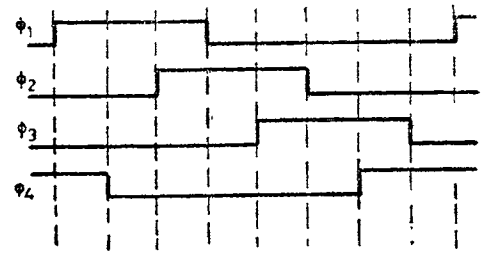


Fig. 4.23. Regimul de comandă al MPP cu jumătate de unghi de pas

Există și posibilitatea realizării unei secvențe de comutație în care pasul unghiular să fie înjumătățit. Pentru aceasta înfășurările sînt comandate cîte două și apoi una singură alternativ: 12—2—23—3—34—4—41—1... sau 14—4—43—3—32—2—21—1 ... Tabelul de adevăr și diagrama de timp din fig. 4.23. sînt de fapt combinații ale tabelelor din fig. 4.21 și 4.22 aparținînd regimurilor de putere întreagă și pe jumătate.

Unghiul de pas cu care se deplasează axul rotorului unui *MPP* poate fi calculat în funcție de p (numărul de poli ai rotorului) și m (numărul de faze) folosind formula: $\alpha_p = \frac{360}{p \cdot m}$ [5]. Pentru exemplul dat în fig. $p=3$ și $m=4$ deci rezultă $\alpha_p=30^0$ iar în regimul cu pas și jumătate $\alpha_p=15^0$. Principiul electronicii de comandă pentru un *MPP* se prezintă ca în fig. 4.24. Blocurile din care se compune apar în fig. 4.25 [6].

Blocul logic pentru stabilirea sensului de rotație emite semnalele *ÎNAINTE* sau *ÎNAPOI* în funcție de nivelul logic al semnalului *SENS*. Frecvența f prescrisă generatorului de tact constituie și frecvența de pas a *MPP*. Numărul de pași care se dorește executat este încărcat în blocul servonumeric și în funcție de sensul de mișcare numărătorul reversibil cu care este dotat acest bloc va fi incrementat sau decrementat. Blocul distribuitor de impulsuri impune secvența de comutație a înfășurărilor *MPP* corespunzător sensului și re-

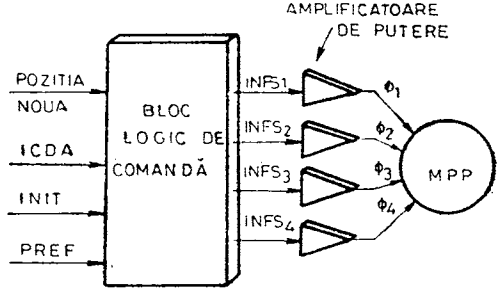


Fig. 4.24. Principiul comenzii electronice a MPP

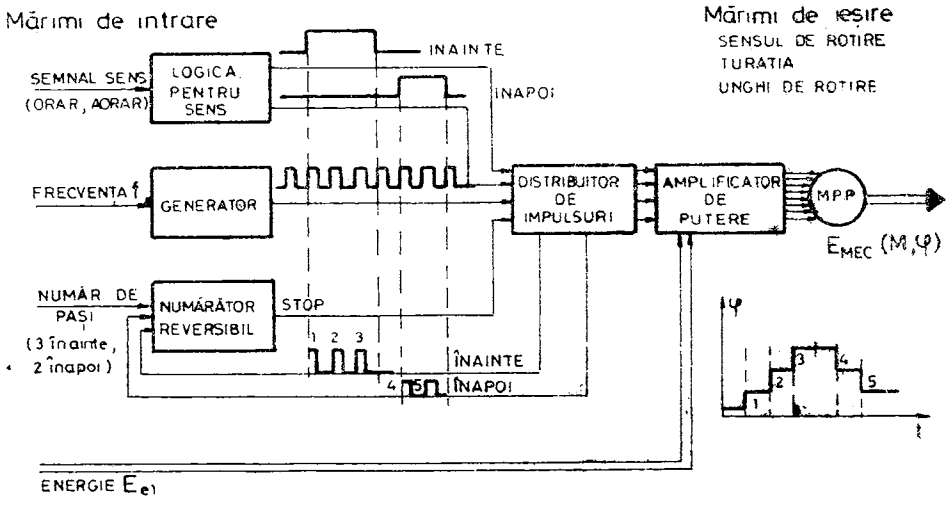


Fig. 4.25. Schema bloc pentru comanda MPP

gimului de comandă alese. În sfîrșit impulsurile furnizate de distribuitor, comandă amplificatoarele de putere care generează impulsurile de curent pentru acționarea *MPP*.

4.4.3. SCHEMA DE COMANDĂ A MPP

O posibilitate de realizare a unei interfețe și a unui bloc de comandă pentru un *MPP*, conectat la microprocesorul 8080, este arătată în fig. 4.26. Un port de 8 bit memorează valoarea poziției noi. Înscrierea octetului se face de către semnalele *I/O.SELMPP*1. Imediat după primirea poziției noi, blocul logic de comandă a *MPP* compară octetul cu poziția curentă aflată în numărător și autorizează generarea impulsurilor de curent pentru *MPP* în sensul în care este nevoie. După efectuarea unui număr de pași semnalul *STOP* execută o tranziție 1—0 care pune semnalul $\overline{INT}$ pe 0 logic. Microprocesorul poate sesiza acest semnal printr-o instrucție de intrare care să genereze *I/OR.SELMPP*2. Aducerea la unu logic a fanionului intern al circuitului 8212 se face cu ocazia înscrierii unui nou octet de poziție sau prin semnalul $\overline{RESET}$.

Același semnal $\overline{RESET}$ permite aducerea servomecanismului în poziția de referință sesizată prin închiderea contactului *PREF*. În acest caz poziția curentă din blocul de comandă al *MPP* va fi zero.

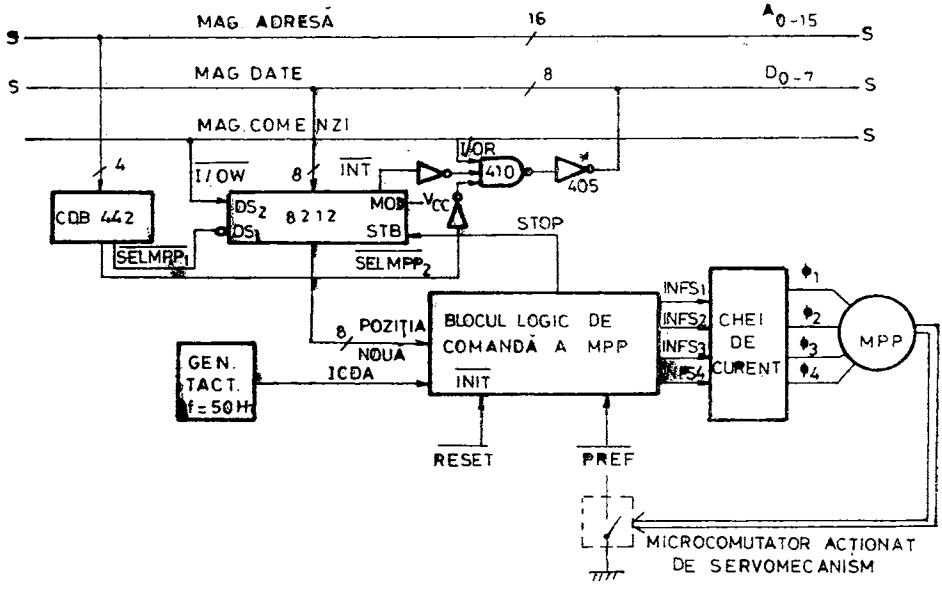


Fig. 4.26. Interfața a comenzii electronice a MPP cu microprocesorul

Schema blocului logic de comandă al *MPP* se împarte în blocul servonumeric și distribuitorul de impulsuri (fig. 4.27). După punerea sub tensiune a sistemului și acționarea comenzii *RESET* bistabilul *RS* este pus pe 0 logic ceea ce provoacă ștergerea numărătoarelor *SN* 74193 și forțarea pe 1 logic a semnalelor $\overline{SENS}$ și *STOP*. Ca urmare MPP începe să deplaseze servomecanismul spre poziția de referință. Atingerea acesteia produce închiderea contactului *PREF* și revenirea bistabilului în starea 1 logic și totodată intrarea schemei în starea normală de funcționare, pe întreg intervalul de lucru de la 00 la FF_{16}.

Ieșirile $A>B$, $A<B$, și $A=B$ ale comparatoarelor contribuie la formarea corectă a semnalelor $\overline{SENS}$, *SENS* și *STOP* în funcție de poziția nouă și cea curentă. Dacă de exemplu *POZIȚIA NOUĂ* are valoarea mai mică decît conținutul numărătorului, semnalul $A<B$ devine 1 logic și deci la intrarea *ECDA* apar semnalele de tact care vor incrementa conținutul acestora pînă cînd $A=B$ devine 1 logic, moment în care semnalul *STOP* blochează impulsurile *ICDA* și *MPP* se oprește.

MPP se va menține în această poziție pînă cînd microprocesorul nu modifică valoarea octetului din registrul tampon de ieșire.

Schema distribuitorului de impulsuri este prevăzută să genereze o secvență de comutație caracteristică regimului de putere completă. Funcționarea are loc

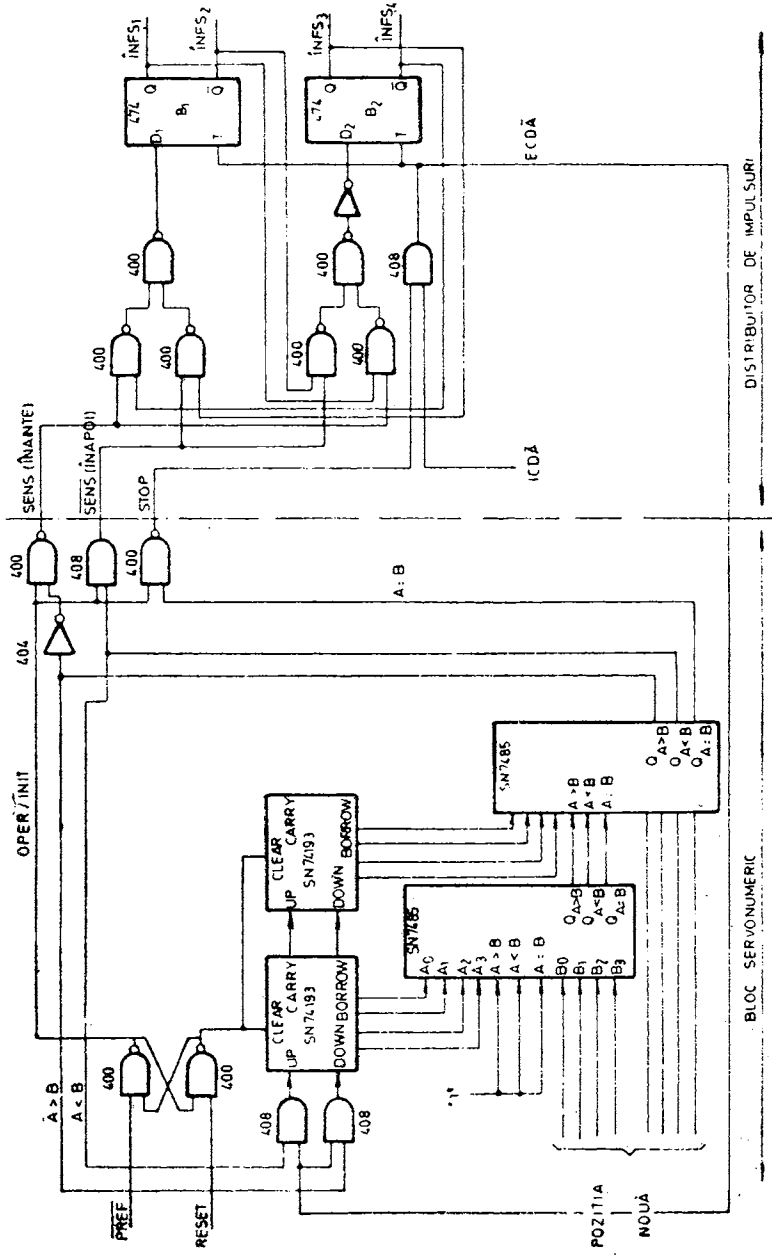


Fig. 4.27. Schema blocului logic de comandă a MPP

conform organigramei stărilor din fig. 4.28, în care stările A, B, C, D, corespund celor 4 paşi ai secvenţei de comutaţie. Observînd că unele coloane din tabelul de adevăr din fig. 4.22 sînt complementare ($INFS_1=\overline{INFS_3}$ şi $INFS_2=\overline{INFS_4}$) rezultă o simplificare a logicii de generare a semnalelor de ieşire $INFS_{1-4}$.

Pe de altă parte cele patru stări din fig. 4.28 pot fi codificate cu două bistabile. Deoarece bistabilele dispun de ieşiri complementare semnalele $INFS_{1-4}$ au fost culese la nivelul acestor ieşiri.

Nu este necesar ca semnalul *RESET* să iniţializeze bistabilele $B1$ şi $B2$ pentru că oricare ar fi stările acestora rezultă un vector binar de comandă valabil pentru *MPP*.

Modul de obţinere a ecuaţiilor logice ale intrărilor bistabilelor este arătat în fig. 4.29.

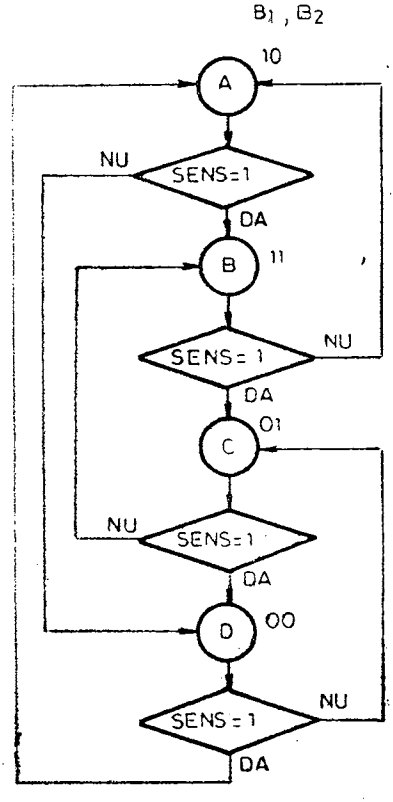


Fig. 4.28. Diagrama stărilor distribuitorului de impulsuri

4.4.4. AMPLIFICATOARELE DE PUTERE

Amplificatoarele de putere sînt destinate furnizării impulsurilor de curent de 550 mA pentru o înfăşurare. De fapt amplificatoarele sînt chei de curent comandate cu nivele logice *TTL* prin semnalele $INFS_1-INFS_4$.

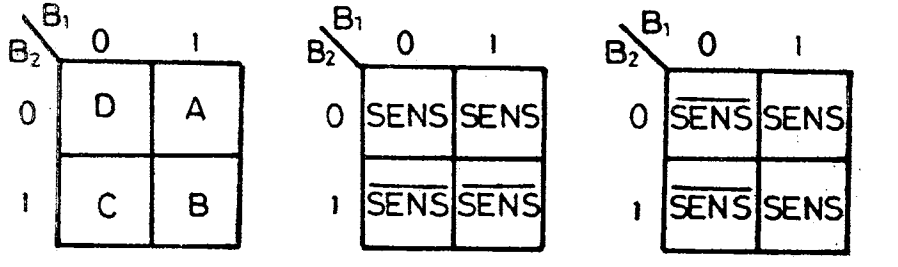


Fig. 4.29. Diagramele pentru minimizarea ecuaţiilor logice ale distribuitorului

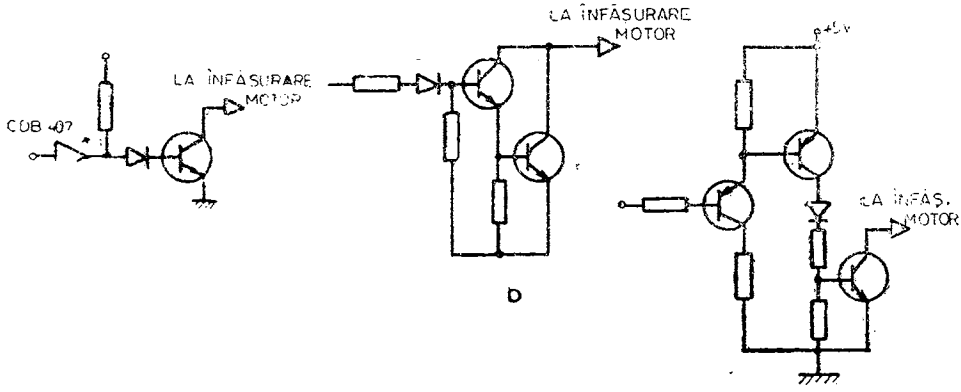


Fig. 4.30. (a, b, c). Variante pentru schema cheii de curent

Cîteva variante de scheme ale cheii de curent pentru puteri diferite sînt arătate în fig. 4.30. Prima schemă (fig. 4.20. a) se folosește în cazul unor motoare de putere mică (dioda protejează circuitele logice care comandă cheia). A doua schemă (fig. 4.30. b) are un montaj Darlington și este potrivită pentru puteri mai mari. În a treia variantă (fig. 4.30. c) etajul de comandă este alimentat la +5 V și tensiunea mai ridicată pentru alimentarea motorului apare numai pe tranzistorul final.

O problemă importantă pentru funcționarea cheii de curent o constituie evitarea supratensiunilor care apar în momentul blocării tranzistorului final. Datorită inductanței înfășurării apare la blocarea tranzistorului fenomenul de autoinducție. Aceasta înseamnă că în înfășurare curentul se menține și după blocarea tranzistorului iar tensiunea autoindusă se însumează celei de alimentare și pune în pericol tranzistorul.

Cîteva soluții pentru evitarea supratensiunilor sînt date în fig. 4.31. Prin introducerea unei diode în paralel cu bobina și avînd polaritatea din figură se asigură o cale de curent prin care curentul de autoinducție dispare mai repede iar supratensiunea are valoarea căderii pe diodă (fig. 4.31. b). Dacă se prevede o rezistență în serie cu dioda se permite apariția unor supratensiuni mai mari dar curentul descrește mai rapid (fig. 4.31. c). În fine o variantă în care în locul rezistenței se introduce o diodă Zener (fig. 4.31, d) limitează supratensiunea la o valoare egală cu tensiunea de alimentare plus tensiunea Zener U_z. Descreșterea va fi tot rapidă dar va avea un caracter liniar [4].

O schemă cu componente dimensionate pentru cheia de curent a unui *MPP* este arătată în fig. 4.32. Rezistența în serie cu înfășurarea motorului reduce constanta de timp a circuitului permițînd o creștere rapidă a curentului. Deoarece rezistența introduce o cădere de tensiune este necesară mărirea tensiunii de alimentare. Pentru o înfășurare avînd rezistența $R_{infs}=9$ ohmi

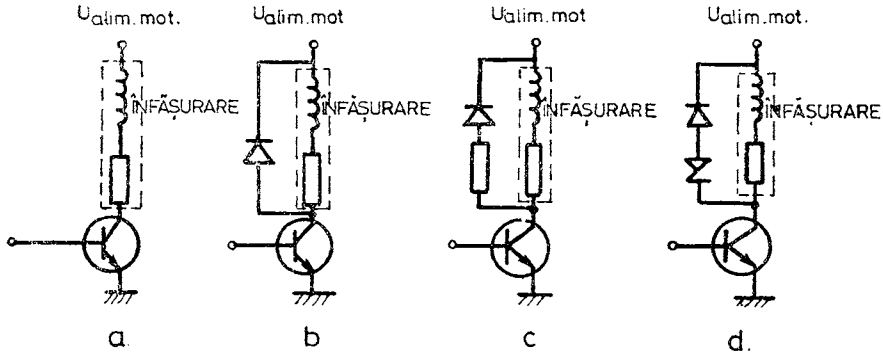


Fig. 4.31. (a, b, c, d) Soluții de evitare ale supratensiunilor pe tranzistorului final

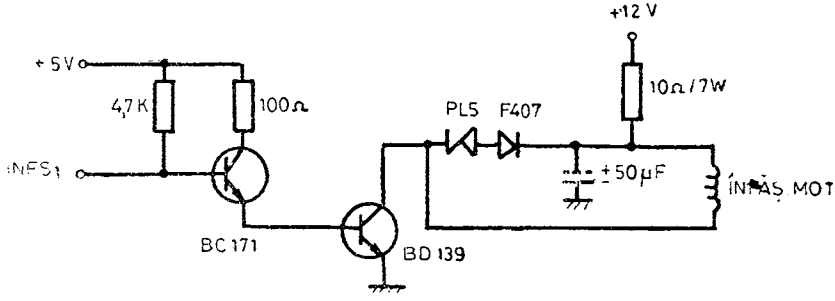


Fig. 4.32. Schema pentru cheia de curent (comanda unei înfășurări a MPP ID-08)

și inductanța $L_{infs}=20$ mH rezultă, luînd rezistența suplimentară de 10 ohmi și frecvența de comandă de 50 Hz, o tensiune de alimentare de 12 V.

Foarte importantă este alegerea tranzistorului final care trebuie să admită cu 30 pînă la 40% mai mult decît curentul maxim prin înfășurare. De asemenea alte criterii de alegere sînt timpii de comutație care trebuie să fie reduși și tensiunile maxime admise pe joncțiunile colector-emitor și colector-bază.

4.4.5. REALIZAREA COMENZII *MPP* PRIN PROIECTARE LOGICĂ PROGRAMATĂ

Folosind conceptul proiectării logice programate funcțiunile îndeplinite de blocul logic de comandă al MPP din fig. 4.27, pot fi executate de microprocesor și programe adecvate. Partea de circuite se reduce astfel foarte mult rămînînd doar cheile de curent și un registru de ieșire pe patru biți (fig. 4.33). Ieșirile circuitului *CDB* 495 asigură comanda logică a cheilor, adică semnalele *INFS*1 — *INFS*4. Înscrierea registrului se face prin semnalele *I/OW*. *SELMPP*,

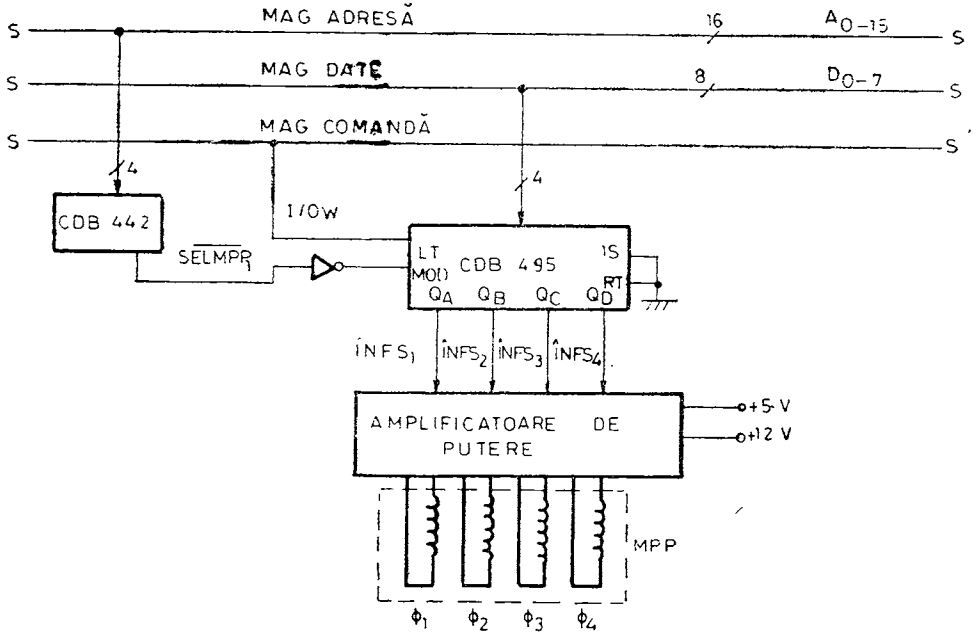


Fig. 4.33. Schema comenzii MPP implementată prin proiectare logică programată

Comutarea curenților prin înfășurările *MPP* este efectuată de programul simplu din tab. 4.2. care pregătește vectorii de comandă corespunzători regimului de putere întreagă.

Tabelul 4.2.

Subprogram de distribuire a impulsurilor pentru MPP

```
;  SUBPROGRAM PENTRU COMANDA MPP GENEREAZĂ O SECVENȚĂ DE COMU-
;  TAȚIE IN REGIM DE PUTERE COMPLETĂ
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         MVI     A,      33H     ;  SE INIȚIALIZEAZĂ VECTORUL
                                 ;  DE COMANDĂ
PASP :   RRC                     ;  ROTIRE VECTOR SPRE DREAP-
                                 ;  TA
         OUT     MPP             ;  SE EXTRAGE VECTORUL
         LXI     B,      0682H   ;  SE ÎNCARCĂ VALOAREA CORES-
                                 ;  PUNZÎND UNUI TIMP=20 MS
TIMP :   DCX     B               ;  DECREMENTEAZĂ REG. BC
         MOV     A,C             ;  TRANSFERA REG. C ÎN ACUMU-
                                 ;  LATOR
         ORA     B               ;  SE EXECUTĂ SAU LOGIC CU B
         JNZ     TIMP            ;  DACĂ Z=0 TRECI LA PASUL
         JMP     PASP            ;  URMĂTOR
```

Tabelul 4.3.

Subprogram de comandă a MPP

```
; SUBPROGRAMUL COMPARĂ POZIȚIA CURENTĂ (PZC) CU POZIȚIA DORITĂ
; (PZD), DETERMINĂ SENSUL DE MIȘCARE ȘI COMANDĂ REALIZAREA
; NUMĂRULUI DE PAȘI PENTRU CA MPP SĂ AJUNGĂ ÎN POZIȚIA DORITĂ.
; VALOAREA PZD SE AFLĂ ÎN REGISTRUL B. IAR VALOAREA PZC ÎN REGISTRUL C
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SERVO: MOV   A, C    ; SE ÎNCARCĂ VALOAREA POZIȚIEI CURENTE
                     ; ÎN ACUMULATOR
       CMP   B       ; SE COMPARĂ CELE DOUĂ VALORI
       JZ    TERM    ; DACĂ SÎNT EGALE MOTORUL NU MIȘCĂ
       JNC   DECR    ; DACĂ PZC>PZD SALT LA DECREMENTARE
INCR:  CALL  PASP    ; ÎN CAZ CONTRAR SE EXECUTĂ UN PAS
                     ORAR
       INR   C       ; SE MĂREȘTE CU UNU PZC
       JMP   SERVO   ; SE REIA COMPARAȚIA PZC CU PZD
DECR:  CALL  PASN    ; SE EXECUTĂ UN PAS ANTIORAR
       DCR   C       ; SE MICȘOREAZĂ CU UNU PZC
       JMP   SERVO   ; SE REIA COMPARAȚIA PZC CU PZD
TERM:  RET           ; REVINE ÎN PROGRAMUL PRINCIPAL
; PROGRAMUL PENTRU PAS ORAR. SE PRESUPUNE LOCAȚIA VEC INIȚIALI-
  ZATĂ CU 33(16).
PASP:  LDA   VEC     ; SE ÎNCARCĂ VECTORUL DE COMANDĂ
       RRC           ; SE PREGĂTEȘTE COMANDA SENS ORAR
       STA   VEC     ; SE MEMOREAZĂ ULTIMA COMANDĂ
       OUT   MPP     ; SE EXTRAGE COMANDA, MPP FACE UN PAS
       CALL  TIMP    ; INTERVAL DE TIMP ÎNTRE DOI PAȘI
       RET
; SUBPROGRAM PENTRU PAS ANTIORAR, SE PRESUPUNE LOCAȚIA VEC INIȚIA-
; LIZATĂ CU 33(16).
PASN:  LDA   VEC     ; SE ÎNCARCĂ VECTORUL DE COMANDĂ
       RLC           ; SE PREGĂTEȘTE COMANDA SENSANTIORAR
       STA   VEC     ; SE MEMOREAZĂ ULTIMA COMANDĂ
       OUT   MPP     ; SE EXTRAGE COMANDA. MPP FACE UN PAS
       CALL  TIMP    ; INTERVAL DE TIMP ÎNTRE DOI PAȘI
       RET
```

Prin încărcarea în acumulator a valorii 00110011 (33_{16}) și rotirea conținutului acumulatorului se obține un nou vector de comandă care este extras prin instrucția *OUT MPP*. Între doi pași ai motorului se așteaptă într-o buclă scurgerea a cca 20 ms ceea ce asigură o frecvență de pas de 50 pași/s.

Schimbarea sensului de mișcare al *MPP* nu se poate face decît prin introducerea instrucției *RLC* în locul instrucției RRC.

Un alt program tab. 4.3 realizează funcțiunile blocului servonumeric prezentat în paragraful 4.4.3. Acest program primește pe calea registrului *B* ca parametru POZIȚIA NOUĂ. Valoarea e memorată și apoi se compară cu

valoarea poziției curente. În funcție de rezultatul comparației se trece la executarea unui pas prin apelarea unuia din subprogramele pentru realizarea unui pas arătate mai sus și apoi se actualizează conținutul poziției curente. După atingerea egalității între poziția nouă și cea curentă se revine în programul principal.

BIBLIOGRAFIE

1. *TK-80 User's Manual*, NIPPON ELECTRIC Co., Ltd, 1977.
2. A. Lesea: R. Zoks: *Microprocessor Interfacing Techniques*, SYBEX Inc., 1978.
3. E. Petriu: *N-User Asynchrono Us Arbiter*, COMPUTER DESIGN, August 1980, p. 126—132.
4. Paul Giacomo,: *A Stepping Motor Primer—Part 1*; *Theory of Operation*, BYTE February 1979, Vol. 4, No. 2, p. 90—105.
5. A. Kelemen; M. Crivii: *Motoare electrice pas cu pas*, Ed. Tehnică, 1975.
6. * * *: *Taschenbuch Elektrotechnik*, Band 3, Verlag Technik, Berlin 1978.

CAPITOLUL 5

CONECTAREA UNOR PERIFERICE ANALOGICE

Una din raţiunile popularităţii microprocesoarelor o constituie capacitatea lor, avantajoasă din punct de vedere al raportului performanţe/cost, de a fi implementate în diverse domenii ale activităţii umane. Marea majoritate a acestor aplicaţii se referă la o realitate înconjurătoare mai degrabă analogică decît numerică. Din acest motiv este evidentă importanţa deosebită a problemelor conectării bidirecţionale a microprocesoarelor la lumea analogică. În ultimă instanţă, rezolvarea acestor probleme se rezumă, din punct de vedere al „hardware"-ului, la conectarea la microprocesor a unor convertoare analog-numerice (*CAN*) şi numeric-analogice (*CNA*) pentru realizarea achiziţiei şi respectiv distribuţiei de date.

5.1. PROBLEME GENERALE ALE ACHIZIŢIEI DE DATE [1], [2], [3]

Achiziţia de date, rezultate prin conversii analog-numerice, are ca scop : memorarea, transmiterea, prelucrarea şi afişarea unor valori numerice caracteristice ale acestora. Folosirea tehnicilor numerice face posibilă atingerea acestor obiective în condiţii economice avantajoase.

Elementul esenţial al oricărui sistem de achiziţii de date este convertorul analog-numeric. Tehnologia actuală a făcut posibilă realizarea unor convertoare analog-numerice integrate ieftine, perfect compatibile cu microprocesoarele (de exemplu convertorul *AD*7574 cu o rezoluţie de 8 biţi, 15 μs timp de conversie şi avînd posibilitatea de a fi tratat ca o locaţie de memorie *RAM*, *ROM* sau memorie lentă).

La stabilirea unei anumite configurații a sistemului de achiziții de date trebuie să se țină seama de cîțiva factori esențiali : rezoluția și precizia convertoarelor, frecvența de eșantionare, capacitatea sistemului de prelucrare în timp real a datelor, condiții impuse de caracteristicile semnalelor de intrare, numărul de canale analogice care vor fi monitorizate și, nu în ultimul rînd ca importanță, costul sistemului.

Din punct de vedere al rezoluției convertoarele analog-numerice integrate actuale oferă 8, 10 sau 12 biți iar cele hibride ajung la 13, 14 biți. Se garantează în mod curent de către producători, precizii de neliniaritate de $\pm 1/2$ din valoarea bitului cel mai puțin semnificativ.

Eșantionarea și memorarea temporară a valorii eșantioanelor precede conversia propriu-zisă. Frecvența de eșantionare este funcție de banda semnalelor de intrare, de parametrii acestor semnale care sînt luați în considerare, precizia impusă, viteza convertoarelor analog-numerice. Astfel o frecvență de eșantionare egală cu dublul frecvenței armonicii superioare din componența spectrală a semnalului de intrare poate fi suficientă în cazul în care interesează parametrii statistici ai acestui semnal. În cazul în care se pune însă problema reconstrucției semnalului de intrare e de dorit ca frecvența de eșantionare să fie cel puțin de 8—10 ori mai mare decît frecvența armonicii superioare. Eroarea finală de prelucrare numerică a semnalelor cumulează componente datorate atît eșantionării cît și cuantizării. Rezultă deci dependența dintre rezoluția convertoarelor, frecvența de eșantionare și precizia de prelucrare. Convertoarele analog-numerice curente reclamă perioade de eșantionare de 25 μs sau 15 μs, ajungîndu-se în unele cazuri la 4 μs sau 1 μs. Din punct de vedere al circuitelor de eșantionare și memorare este de preferat să se impingă frecvența de eșantionare spre limita maximă admisă de convertoare și procesor. Această tendință se datorește faptului că precizia elementelor de memorare a eșantioanelor scade odată cu creșterea duratei de memorare. Sînt situații, în cazul unor semnale de bandă joasă, cînd se renunță la eșantionare și memorare recurgîndu-se direct la conversia analog-numerică.

Capacitatea procesorului de prelucrare în timp real influențează pe de o parte performanțele sistemului, impunîndu-se un compromis între banda semnalelor de intrare și complexitatea algoritmilor de prelucrare în timp real, iar pe de altă parte influențează asupra structurii sistemului. Astfel în cazul creșterii benzii semnalelor și al volumului de calcule de efectuat se impune alegerea unor procesoare mai rapide („bit-slice") sau a unor structuri multiprocesor, utilizarea *DMA*, sau utilizarea unor dispozitive aritmetice rapide etc.

Caracteristicile semnalelor de intrare pot impune utilizarea unor tehnici speciale de conversie : polarizarea domeniului, comutarea automată a domeniilor, compresia logaritmică, filtrarea semnalelor de intrare. De obicei aceste tehnici speciale, de condiționare a semnalelor, se aplică înainte de eșantionare.

În funcție de numărul canalelor analogice care vor fi monitorizate există sisteme de achiziții cu unul sau mai multe canale. Monitorizarea mai multor

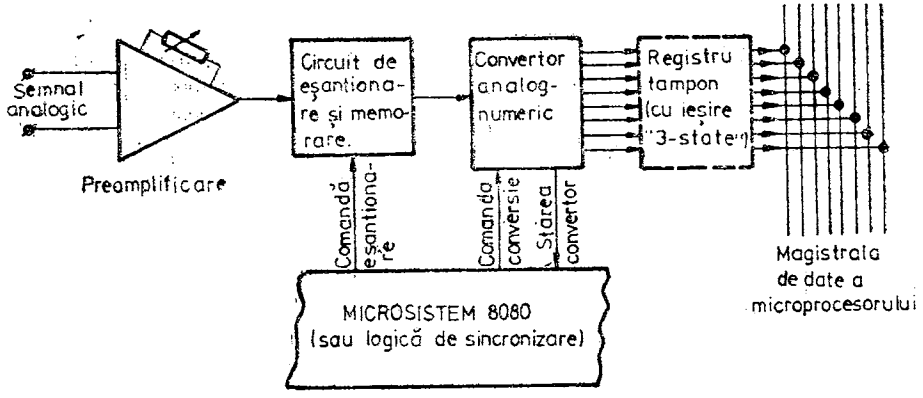


Fig. 5.1. Sistem unicanal pentru achiziţii de date

canale se poate realiza pe diferite căi : multiplexarea valorilor numerice obţinute după conversie, multiplexarea semnalelor analogice înainte de eşantionare, multiplexarea analogică a eşantioanelor.

În fig. 5.1 se prezintă structura unui sistem de achiziţii cu un singur canal. În general convertoarele analog-numerice necesită un anumit timp pentru realizarea conversiei. Datorită acestui fapt, aşa cum se vede şi în figură, convertorul este precedat de un circuit de eşantionare şi memorare. Acest circuit prelevează cîte un eşantion din semnalul analogic şi memorează eşantionul pe toată durata conversiei. Utilizarea circuitului de eşantionare şi memorare permite stabilirea precisă a momentelor de timp la care se face prelevarea eşantioanelor care se măsoară. Atît circuitul de eşantionare şi memorare cît şi convertorul analog-numeric operează sub controlul unor semnale exterioare. Semnalele de control declanşează eşantionarea şi apoi conversia. Deoarece circuitele de eşantionare şi memorare au un factor de amplificare unitar şi o eroare absolută independentă de valoarea eşantionului este necesară utilizarea prealabilă a unui preamplificator. Preamplificarea semnalului care se eşantionează asigură o reducere a erorilor relative de eşantionare şi cuantizare ; eroarea relativă fiind egală cu raportul dintre eroarea absolută (care este constantă) şi valoarea semnalului măsurat (valoare ce poate fi mărită prin preamplificare). Preamplificarea constituie o condiţionare a semnalului de intrare.

În fig. 5.2 se prezintă structura unui sistem de achiziţii de date cu multiplexarea valorilor numerice obţinute după conversie. Structura constă în repetarea pentru fiecare canal a structurii unicanal din figura precedentă.

Tehnologia actuală pune la dispoziţie convertoare analog-numerice cu ieşiri „3-state", astfel încît multiplexarea acestor convertoare se reduce la simpla legare în paralel a ieşirilor lor la magistrala de date a microprocesorului.

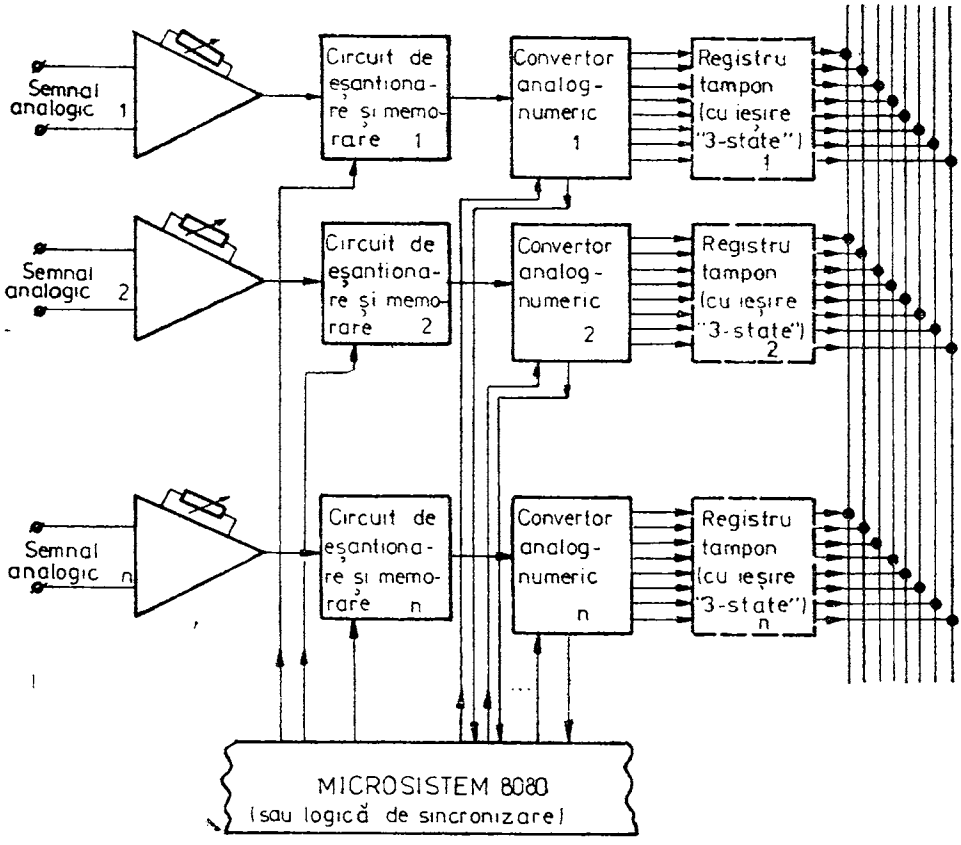

Semnal analogic 1
Circuit de eşantionare şi memorare 1
Convertor analog-numeric 1
Registru tampon (cu ieşire "3-state") 1
Semnal analogic 2
Circuit de eşantionare şi memorare 2
Convertor analog-numeric 2
Registru tampon (cu ieşire "3-state") 2
Semnal analogic n
Circuit de eşantionare şi memorare n
Convertor analog-numeric n
Registru tampon (cu ieşire "3-state") n
MICROSISTEM 8080
(sau logică de sincronizare)

Fig. 5.2. Sistem multicanal de achiziţii de date, cu multiplexare numerică

Structura descrisă este avantajoasă în cazul sistemelor industriale de achiziţii de date, care monitorizează puncte de măsurare dispersate pe o arie mai mare. Conversia analog-numerică la faţa locului pentru fiecare mărime analogică prezintă avantajul creşterii imunităţii la zgomot a datelor transmise spre procesorul de monitorizare, fiind posibilă chiar folosirea în acest scop a unor coduri detectoare şi/sau corectoare de erori.

De asemenea este posibilă eşantionarea simultană a tututor semnalelor de intrare. Preţul de cost scăzut al convertoarelor analog-numerice integrate face perfect realizabilă o astfel de soluţie. Semnalele de sincronizare pot fi generate de microprocesor sau de o logică specială.

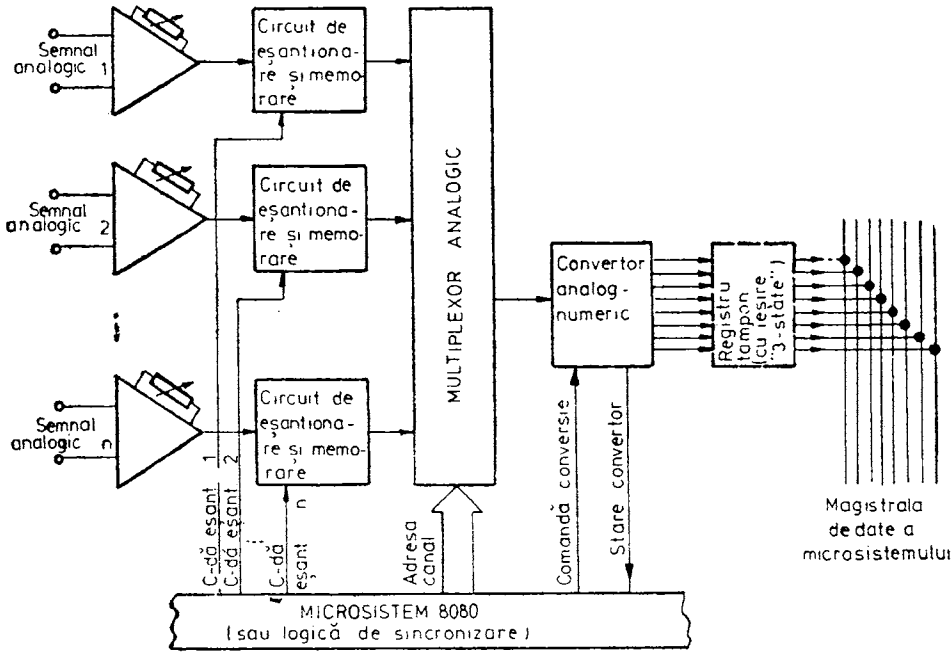


Fig. 5.3. Sistem de achiziţii de date cu multiplexarea eşantioanelor analogice

Un alt tip de sistem multicanal pentru achiziţii de date îl constituie cel în care sînt multiplexate analogic ieşirile circuitelor de eşantionare şi memorare. În fig. 5.3 se prezintă structura unui astfel de sistem.

Structura este mai economică decît precedenta necesitînd un singur convertor analog-numeric pentru toate canalele. Şi în cazul acestei structuri este posibilă eşantionarea simultană a tuturor canalelor de intrare. Este o structură care se pretează bine la eşantionări de frecvenţă ridicată (timpul de memorare al eşantioanelor analogice fiind un parametru critic, lungimea sa fiind dificil de realizat). Ca şi în cazul structurii precedente multiplexarea poate fi secvenţială sau aleatoare. Semnalele de control pot fi asigurate de microprocesor sau de o logică specială.

În fig. 5.4 se prezintă o structură şi mai simplă de sistem pentru achiziţii de date utilizînd multiplexarea semnalelor analogice înainte de eşantionare. Se utilizează un singur circuit de eşantionare şi memorare şi un singur convertor analog-numeric pentru toate canalele de intrare. Ca şi în toate celelalte struc-

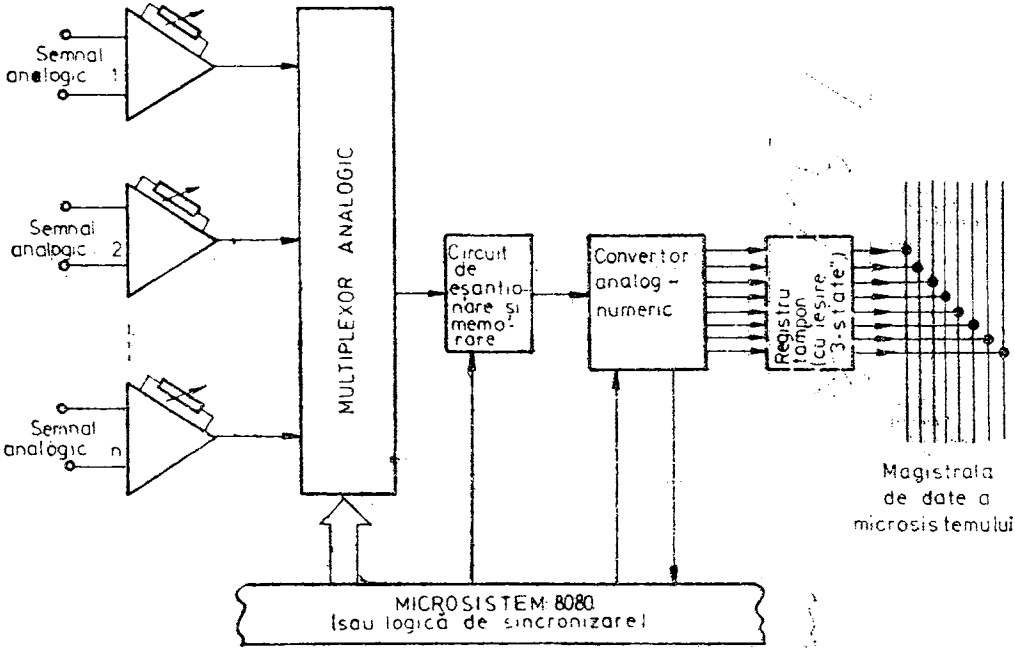


Fig. 5.4. Sistem de achiziţii de date cu multiplexarea semnalelor analogice înainte de eşantionare

turi precedente sincronizările pot fi controlate de către microprocesor sau de o logică special destinată acestui scop. Multiplexarea poate fi alcătoare sau secvenţială.

Prin suprapunerea ciclului de memorare cu ciclul de selectare a următorului canal se poate obţine o creştere a vitezei de achiziţie a sistemului. Totuşi această ultimă structură rămîne cea mai lentă dintre toate structurile multicanal pentru achiziţii de date.

Costul sistemului de achiziţii de date se poate reduce în primul rînd prin alegerea unor componente de sistem cît mai ieftine dar cu performanţe acceptabile. Utilizarea microprocesoarelor în sistemele de achiziţii de date oferă încă o posibilitate suplimentară de reducere a costului echipamentului „hardware" prin implementarea prin programe a unora dintre funcţiunile acestui echipament. Evident trebuie să se ţină seama pe lîngă o reducere a vitezei de prelucrare în timp real şi de costul programării.

5.2. METODE DE CONECTARE A CONVERTOARELOR ANALOG-NUMERICE LA MICROPROCESOR

O problemă importantă care apare la conectarea convertoarelor analog-numerice (*CAN*) la microprocesor este cea a sincronizării dintre ele, ambele dispozitive avîndu-şi tactul lor propriu. Alegerea uneia sau alteia dintre metodele cunoscute de legare a perifericelor la microprocesoare, se face pe baza relaţiei dintre timpul de conversie al *CAN* şi ciclul microprocesorului.

5.2. CONECTAREA CONVERTOARELOR ANALOG-NUMERICE RAPIDE

Convertoarele analog-numerice rapide se caracterizează printr-un timp de conversie de ordinul a cîtorva cicluri de microprocesor. Cea mai simplă metodă de conectare a unor astfel de convertoare la microprocesor constă în tratarea lor ca adrese de memorie. Declanşarea conversiei se face printr-o instrucţiune cu referire la memorie, urmată de cîteva instrucţiuni fără nici un alt efect decît consumarea timpului necesar pentru efectuarea conversiei iar în final să se citească rezultatul conversiei printr-o altă instrucţiune convenabilă.

În fig. 5.5 se prezintă principiul de interfaţare a unui *CAN* rapid la un microprocesor 8080. Convertorul este tratat ca o adresă de memorie. Declan-

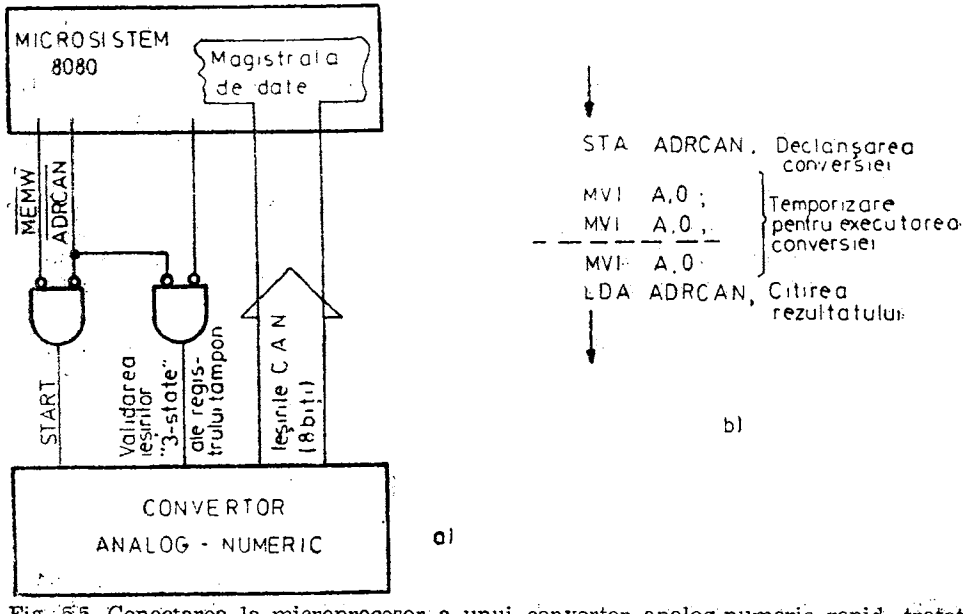
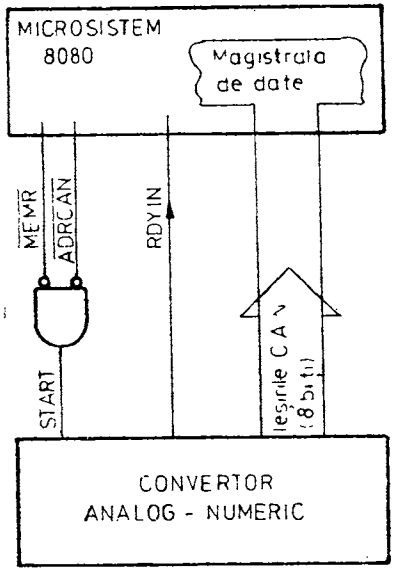
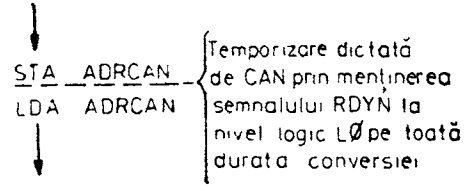


Fig. 5.5. Conectarea la microprocesor a unui convertor analog-numeric rapid, tratat ca o adresă de memorie (a) şi segmentul de program pentru realizarea unei conversii



a)



b)

Fig. 5.6. Conectarea la microprocesor a unui convertor analog numeric rapid, tratat ca o adresă de memorie lentă (a) şi segmentul de program corespunzător unei conversii (b)

şarea conversiei se face printr-o instrucţie de înscriere în memorie la adresa *ADRCAN* alocată convertorului : *STA ADRCAN.*

Consumarea timpului necesar pentru efectuarea conversiei este asigurată prin repetarea de cîte ori este necesar, a instrucţiei *MVI A*, 0 (care consumă cîte 7 stări). Rezultatul conversiei este citit şi introdus în acumulator prin-instrucţia *LDA ADRCAN.*

O altă posibilitate de conectare la microprocesor a unui CAN rapid o reprezintă cea de tratare a sa ca memorie lentă. În fig. 5.6.a se prezintă modul de realizare a interfeţei *CAN* cu microprocesorul 8080 iar în fig. 5.6. b segmentul de program pentru realizarea unei conversii în cazul utilizării acestei metode de conectare.

CAN este tratat ca o adresă, *ADRCAN*, de memorie. Conversia este declanşată la începutul instrucţiei *LDA ADRCAN* de citire şi încărcare în acumulator a conţinutului adresei *ADRCAN* de memorie.

5.2.2. CONECTAREA CONVERTOARELOR ANALOG-NUMERICE LENTE

În cazul convertoarelor analog-numerice lente nu mai pot fi practic utilizate metodele de conectare la procesor descrise în paragraful precedent, deoarece s-ar consuma prea mult din timpul de calcul numai pentru a se aştepta

terminarea conversiilor. O modalitate mai eficientă de conectare a *CAN* lente este cea prin care microprocesorul declanşează procesul de conversie analog-numerică şi revine la programul principal executînd alte calcule în care nu este necesar rezultatul conversiei în curs. Convertorul se conectează astfel încît să poată cere o întrerupere atunci cînd a terminat de efectuat conversia. Întreruperea cerută de convertor va determina saltul la un subprogram de tratare a conversiei.

În fig. 5.7 se prezintă modul de conectare a unui *CAN* lent la microprocesor făcîndu-se uz de posibilitatea de întrerupere a microprocesorului. Pentru generarea semnalului de întrerupere se foloseşte semnalul $\overline{BUSY}$ care marchează (atunci cînd este la nivelul 0) intervalul de timp în care *CAN* este ocupat cu executarea conversiei curente. Întreruperea trebuie să se producă doar la terminarea acestui semnal. Din acest motiv este necesară mascarea acestui semnal de întrerupere pînă în momentul terminării conversiei. Deoarece microprocesorul 8080 nu dispune de instrucţii de mascare şi demascare selectivă a întreruperilor este necesar ca această operaţie să fie efectuată prin circuite speciale anexate convertorului.

Conversia este declanşată prin instrucţia *STA ADRCAN*, unde *ADRCAN* este adresa convertorului. Următoarea instrucţiune, EI, validează sistemul de întreruperi al micoprocesorului. Programul principal continuă cu alte instrucţii pînă în momentul în care apare semnalul *INT*. Apariţia acestui semnal în condiţiile validării sistemului de întreruperi determină saltul la suprogramul de tratare a întreruperii cerute de *CAN*. Prima instrucţie a acestui subprogram, *DI*, invalidează sistemul de întreruperi al microporocesorului astfel încît orice altă eventuală cerere de întrerupere, provenind de la oricare periferic, să nu fie luată în considerare pînă la terminarea subprogramului. Prin instrucţiunea *LDA ADRCAN* se citeşte şi se încarcă în acumulator rezultatul conversiei.

O altă metodă des utilizată pentru conectarea *CAN* lente la microprocesoare o constituie controlul periodic prin program al stării convertoarelor. În momentul depistării faptului că s-a terminat conversia se trece la transferul de date dinspre *CAN* spre microprocesor. Soluţia este deosebit de simplă din punct de vedere al „hardware"-ului nemaifiind necesare circuite speciale pentru întreruperi. Ca dezavantaj trebuie însă menţionat faptul că se pierde din timpul de calcul pentru testarea indicatorilor de stare ai convertoarelor. În fig. 5.8 se prezintă acest mod de conectare şi organigrama de principiu a programului corespunzător.

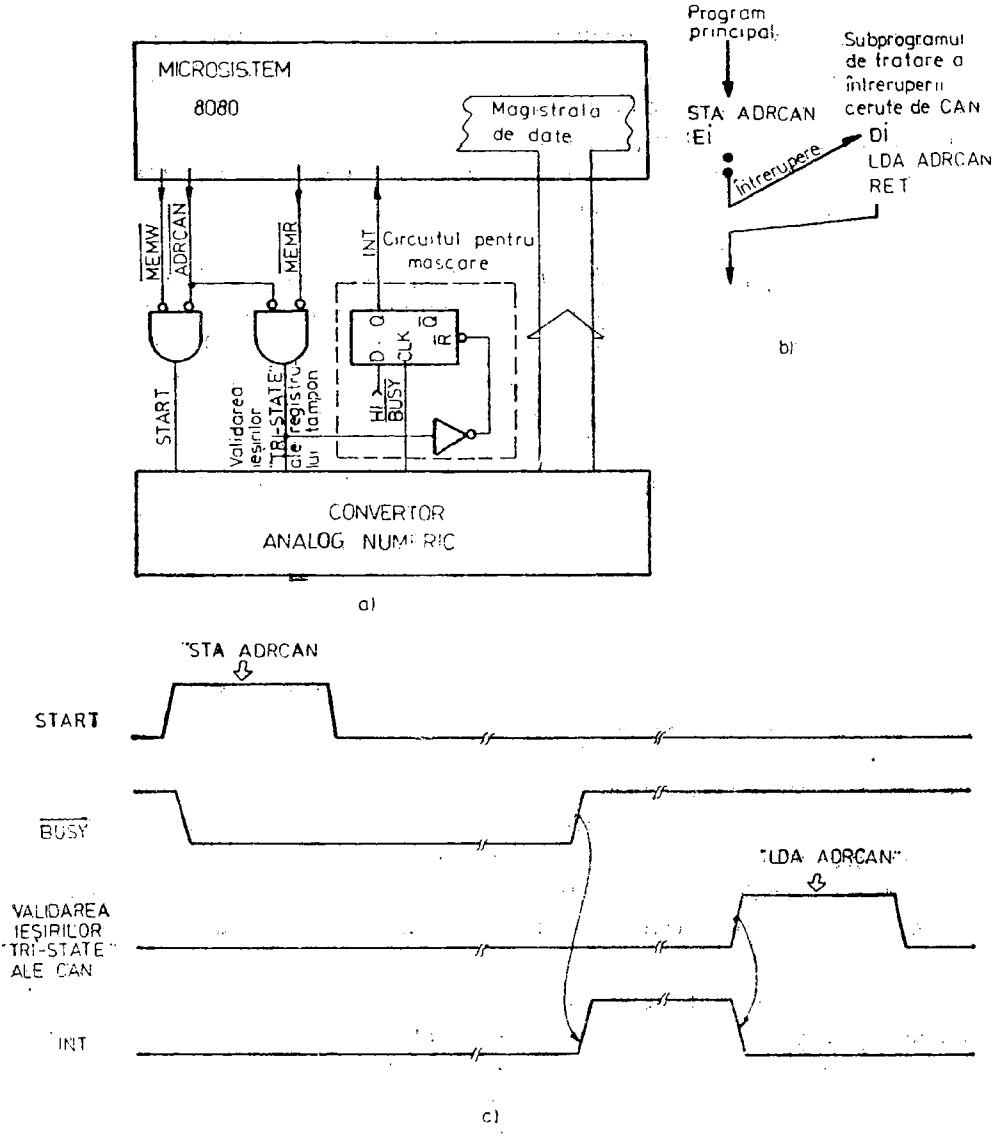
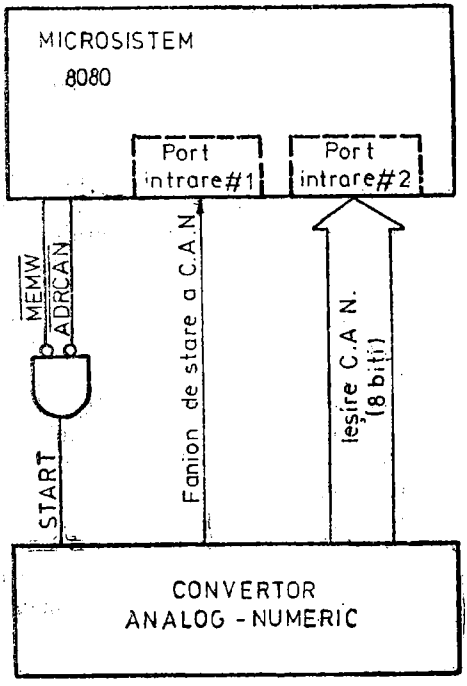


Fig. 5.7. Conectarea la microprocesor a unui CAN lent: a) structura interfeței cu circuitul pentru mascarea și demascarea întreruperilor; b) programul de conversie; c) diagramele de timp descriind comportarea circuitului de mascare și demascare



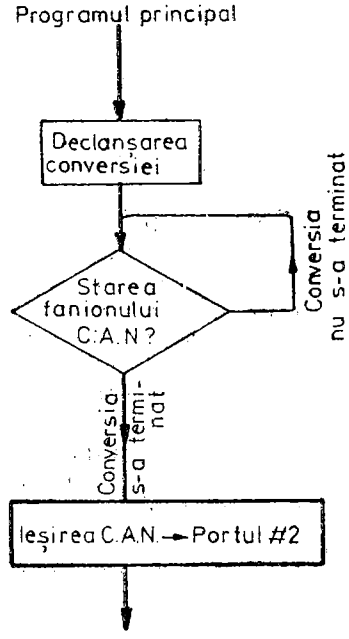


Fig. 5.8. Conectarea la microprocesor a unui CAN lent folosindu-se controlul prin program a stării convertorului

Pentru a se mări eficiența achiziției de date în cazul folosirii controlului prin program a stării *CAN* se poate renunța la controlul continuu al stării *CAN* pînă la depistarea terminării conversei. O parte din timpul disponibil din momentul declanșării conversiei pînă la terminarea ei, este utilizat pentru eventuale calcule intermediare apelîndu-se, din cînd în cînd numai, un subprogram de testare a *CAN*. Această variantă a metodei de control prin program este avantajoasă îndeosebi în cazul *CAN* foarte lente. În fig. 5.9 se prezintă organigrama de principiu pentru varianta descrisă.

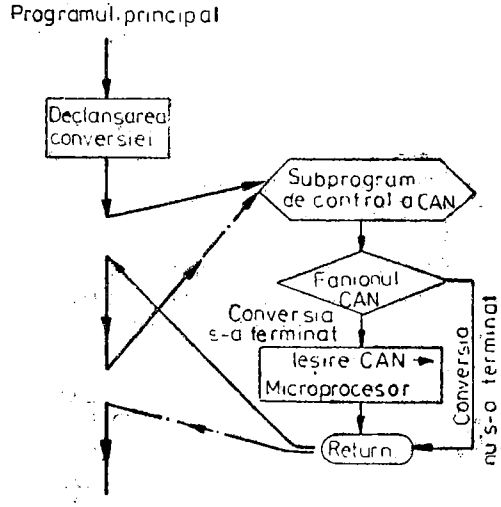


Fig. 5.9. Controlul prin program, cu intermitențe, a stării CAN

5.2.3. CONECTAREA UNUI CONVERTOR ANALOG-NUMERIC DE 10 BIȚI (AD571) [4]

Deseori se pune problema interfațării la microprocesorul 8080 a unor *CAN* avînd rezoluția mai mare de 8 biți. În acest caz citirea se face în două etape : întîi se citește octetul cel mai semnificativ apoi octetul mai puțin semnificativ. În prelucrările ulterioare, evident, se va lucra cu aceste rezultate ca operanzi avînd lungimea de doi octeți.

În fig. 5.10 se prezintă modul de conectare la un microsistem 8080 a unui convertor analog-numeric rapid *AD* 571 avînd 10 biți de ieșire. Convertorul este tratat ca o adresă de memorie conform procedurii prezentate în fig. 5.5.

Convertorul digitalizează eșantioanele memorate de circuitul integrat „de eșantionare-memorare" *AD* 582. Acest circuit necesită un timp de achiziție de 10 μs utilizînd o capacitate de memorare C_H=300 pF. Căderea de tensiune în faza de memorare este mai mică de 100 pV/ms.

Circuitul „de eșantionare și memorare" constituie de asemenea și un circuit tampon cu impedanță de intrare mare pentru convertorul analog-numeric.

În registrul de „memorare" al circuitului „de eșantionare și memorare" tensiunea de la ieșirea acestui circuit urmărește orice modificare a tensiunii de alimentare „—V" astfel încît se impune o deosebit de bună stabilizare și filtrare a acestei tensiuni.

Convertorul *AD* 571 necesită un singur semnal de control *BLANC* a $\overline{CONVERT}$. În stare de repaus acest semnal este menținut la nivelul logic 1 iar semnalul de stare a ieșirilor convertorului $\overline{DATA\ READY}$ are de asemenea valoarea logică 1. În stare de repaus ieșirile de date nu sînt disponibile fiind în starea de înaltă impedanță (fig. 5.11).

Trecerea semnalului *BLANC* a *CONVERT* declanșează începerea conversiei eșantionului memorat, proces care durează 25 μs. La terminarea conversiei semnalul $\overline{DATA\ READY}$ generat de convertor trece de la valoarea logică 1 la valoarea 0 iar după maximum 500 ns rezultatul conversiei devine disponibil la cele 10 ieșiri de date, *BIT*1, ... *BIT*10, care trec din starea de înaltă impedanță în starea activă.

Pentru realizarea conversiei, citirea celor 10 biți rezultați în urma conversiei eșantionului și înscrierea în memorie la două adrese succesive a rezultatului conversiei se poate utiliza următorul segment de program :

```
. . . . .
STA  ADRSTART   ;   Declanșarea conversiei.
MVI  A, 0       ;   Presupunînd o durată de 500 ns/ciclu rezultă
MVI  A, 0       ;   o întîrziere de 3,5 μs/instrucție și o tempo-
MVI  A, 0       ;   rizare totală de 7×3,5 μs care împreună cu
MVI  A, 0       ;   stările din faza de aducere a instrucțiunei
MVI  A, 0       ;   LDA ADRØCTETHI acoperă timpul de
```

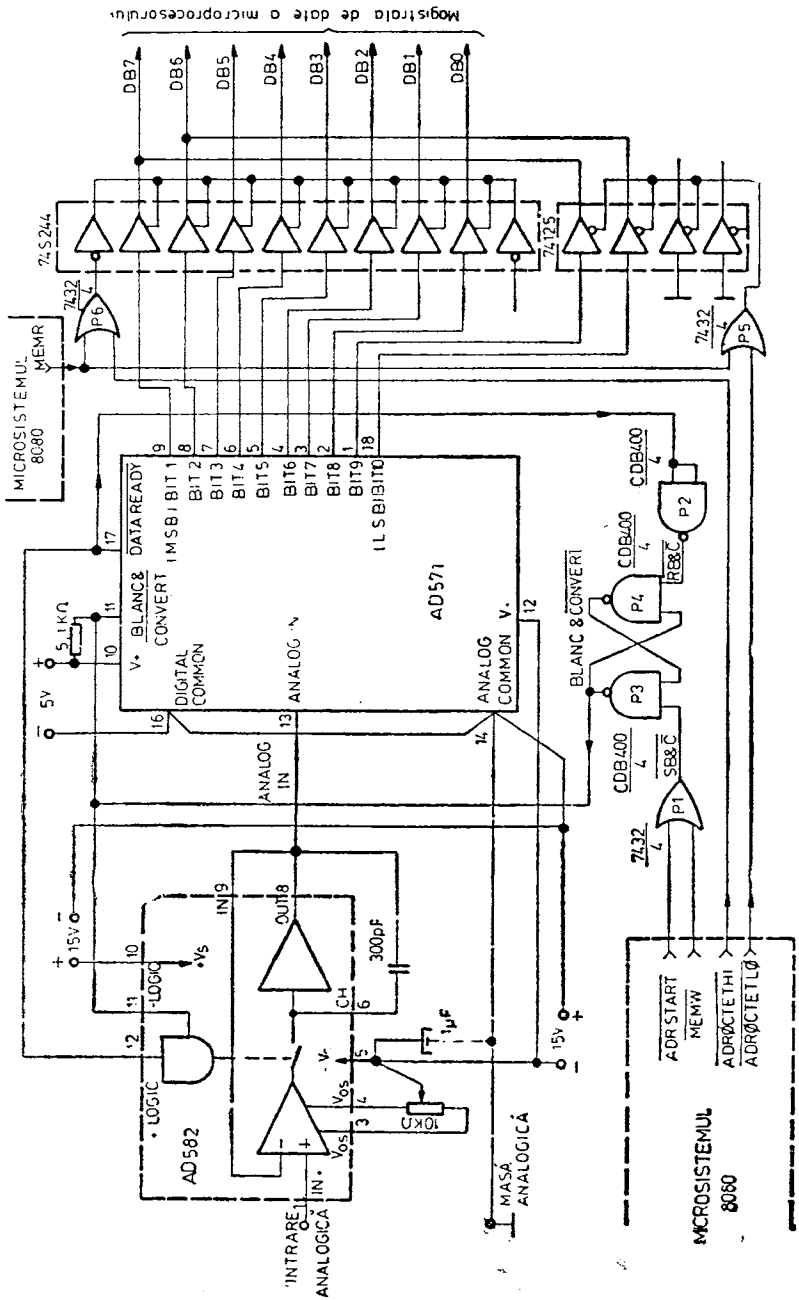


Fig. 5.10. Sistem de achiziții monocanal cu rezoluția de 10 biți, conectat la un microsistem 8080

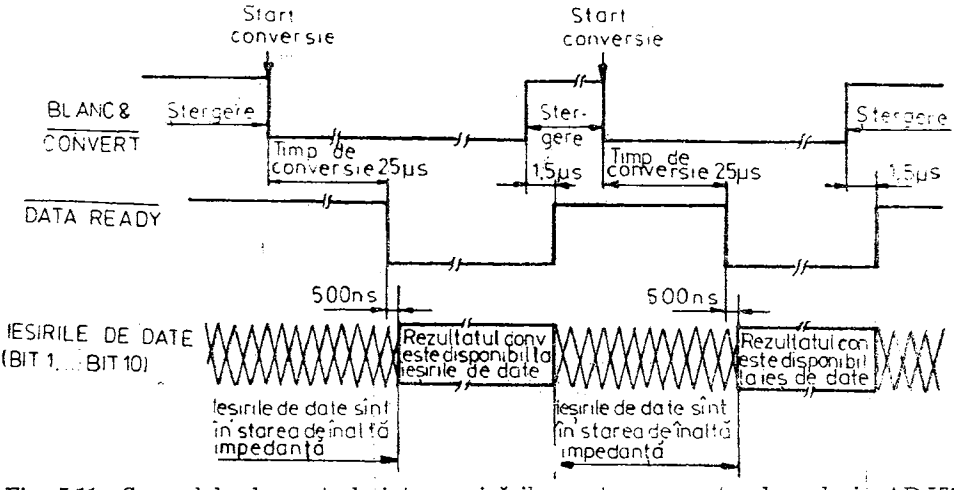


Fig. 5.11. Semnalele de control şi temporizările pentru convertorul analogic AD 571

```
MVI  A, 0      ;
MVI  A, 0      ;
MVI  A, 0      ;
LDA  ADRØCTETHI ;
LXI  H, ESANT  ;
MØV  M, A      ;
LDA  ADRØCTETLØ ;
LXI  H (ESANT+1) ;
MØV  M, A      ;
. . . . .
```

conversie de 25 µs cerut de convertorul *AD* 571.

Se încarcă în acumulator primii 8 biţi mai semnificativi ai rezultatului conversiei eşantionului

Se încarcă în registrele (*H*, *L*) adresa de memorie la care se vor înscrie primii 8 biţi ai rezultatului conversiei eşantionului.

Se înscriu în memoria la adresa *ESANT* primii 8 biţi ai rezultatului conversiei.

Se încarcă în acumulator ultimii 2 biţi mai puţin semnificativi ai rezultatului conversiei.

Se încarcă în registrele (*H*, *L*) adresa de memorie la care se vor înscrie ultimii 2 biţi ai rezultatului conversiei.

Se înscriu în memorie la adresa *ESANT*+1 ultimii 2 biţi ai rezultatului conversiei.

Soluţia adoptată de realizare prin program a întîrzierii necesare pentru efectuarea conversiei deşi dezavantajoasă faţă de soluţia conform fig. 5.6, din punct de vedere al numărului de instrucţii necesare, prezintă avantajul cunoaşterii mai precise a perioadei de eşantionare.

În decursul fazei de execuție a instrucțiunii *LDA ADRØCTETHI* pe magistrala de adrese a microprocesorului apare adresa *ADRØCTETHI* care prin decodificare stabilește nivelul 0 la intrarea $\overline{ADRØCTETHI}$ a porții P6. Acest semnal împreună cu impulsul negativ care apare între timp la cealaltă intrare $\overline{MEMR}$ a porții *P*6 determină validarea porților 3 — stări ce permit accesul ieșirilor *BIT* 1, ..., *BIT* 8 spre magistrala de date a microsistemului de unde sînt încărcate în acumulatorul microprocesorului. Din acumulator octetul citit este încărcat prin instrucțiunea *MØV M, A* în memorie la adresa *ESANT* care a fost în prealabil încărcată în registrul de adrese al microprocesorului.

În decursul execuției instrucțiunii *LDA ADRØCTETLØ* apare după decodificarea adresei un semnal logic 0 la intrarea *ADRØCTETLØ* a porții *P*5. Acest semnal, împreună cu impulsul negativ $\overline{MEMR}$ care apare între timp, validează poarta *P*5 astfel încît următoarele două ieșiri, *BIT*9 și *BIT*10, ale convertorului sînt conectate la magistrala de date a microsistemului de unde sînt încărcate în acumulator. Prin următoarele două instrucții, *LXI H*, (*ESANT*+1) și *MØV M, A*, acești doi biți sînt încărcați în memorie la următoarea adresă față de cea unde au fost încărcați primii 8 biți.

În fig. 5.12 sînt prezentate diagramele de timp care ilustrează fenomenele care au loc în timpul conversiei în cadrul sistemului de achiziții de date avînd structura prezentată în fig. 5.10.

Timpul de 10 μs necesar pentru achiziția următorului eșantion de către circuitul de eșantionare și memorare, este acoperit cu prisosință de către durata de execuție a instrucției *LDA ADRØCTETHI* și a următoarelor instrucții din segmentul de program descris.

5.3. PROBLEME GENERALE ALE DISTRIBUȚIEI ANALOGICE A DATELOR NUMERICE [2]

Microprocesoarele implementate în sisteme de comandă și control au conectate și o serie de periferice analogice : motoare, înregistratoare în coordonate, dispozitive de modulare a undelor, dispozitive de control a luminii, căldurii sau sunetului etc. ... Toate acestea impun existența unor sisteme de distribuție sub formă analogică a datelor numerice de la microprocesor. Aceste sisteme de distribuție au ca element esențial convertorul numeric-analogic.

Alegerea unei anumite configurații a sistemului de distribuție analogică a datelor presupune o analiză atentă a parametrilor impuși : numărul canalelor de ieșire, timpul de stabilizare/canal, viteza de variație a datelor de ieșire,

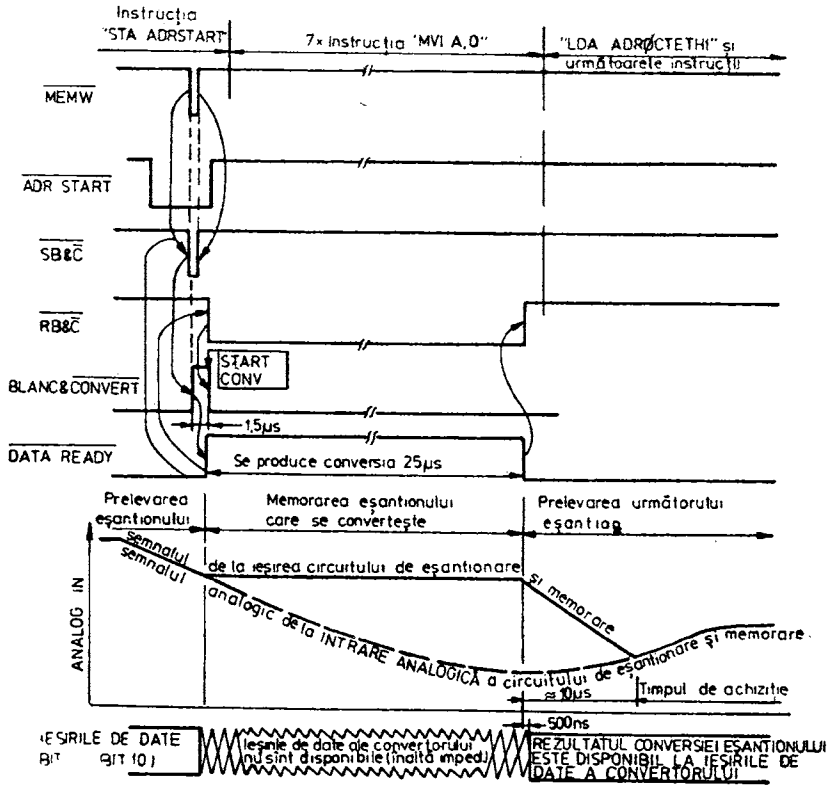


Fig. 5.12. Diagramele de timp ale principalelor semnale din structura sistemului de achiziții de date din fig. 5.12 în decursul procedurii de conversie

rezoluția semnalelor de ieșire, liniaritatea și precizia semnalelor analogice, natura perifericelor analogice care se acționează și, nu în ultimul rînd, costul sistemului.

Există diverse modalități de realizare a operațiilor care au loc într-un sistem de distribuție analogică a datelor. Astfel datele numerice pot fi extrase în serie sau în paralel. Memorarea eșantioanelor pe baza cărora se reconstituie semnalele analogice se poate face în porțiunea numerică sau cea analogică a sistemului de distribuție. Demultiplexarea datelor provenite de la microprocesor se poate face și ea în zona numerică sau cea analogică iar în ultimul caz se poate face cu sau fără utilizarea circuitelor de eșantionare și memorare. Reactualizarea valorilor de ieșire de pe diferitele canale se poate face și ea în

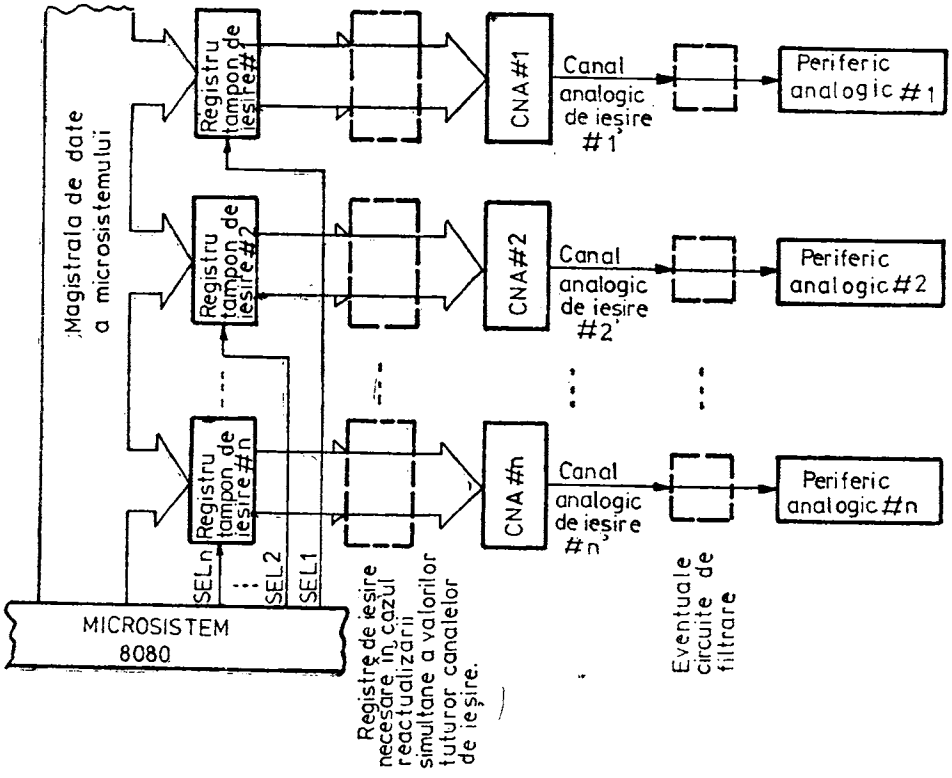


Fig. 5.13. Sistem de distribuţie de date cu demultiplexare digitală şi posibilităţi de reactualizare simultană a valorilor semnalelor de ieşire

mai multe feluri : simultan, secvenţial sau aleator. Conversia numeric-analogică se poate face cu ajutorul unui singur convertor, dacă nu este necesară o reactualizare simultană a semnalelor de ieşire şi dacă se face o demultiplexare analogică, sau cu ajutorul a mai multor convertoare dacă este necesară o reactualizare simultană a valorilor tuturor semnalelor de ieşire şi dacă se face o demultiplexare numerică a datelor ce se distribuie. Modul de prezentare a semnalelor analogice de ieşire poate cere sau nu o filtrare. De asemenea unele canale de ieşire trebuie să fie izolate galvanic de perifericele la care se duc. În figurile 5.13 şi 5.14 se prezintă două tipuri de sisteme de distribuţie de date : cu demultiplexare numerică şi respectiv demultiplexare analogică.

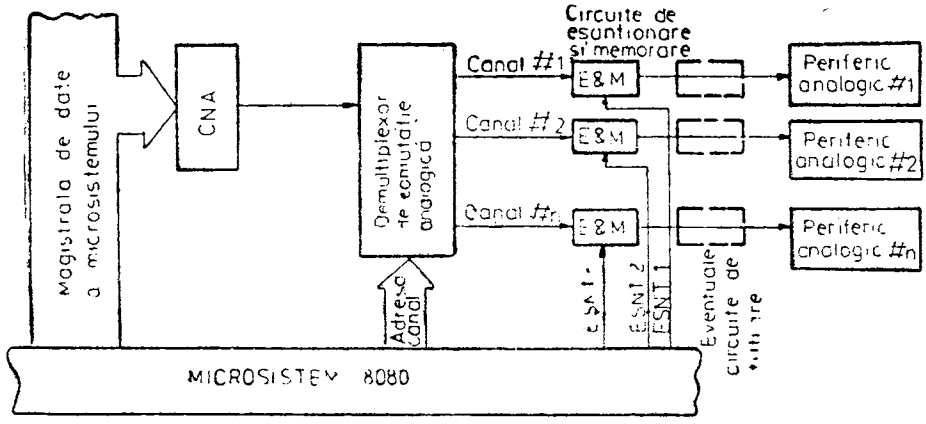


Fig. 5.14. Sistem de distribuție de date cu demultiplexare analogică

5.4. CONECTAREA UNUI ÎNREGISTRATOR ÎN COORDONATE FOLOSIND DOUĂ CONVERTOARE NUMERIC-ANALOGICE

Existența unui înregistrator în coordonate $X-Y$ conectat ca periferic la microsistem, face posibilă extragerea rezultatelor sub formă de grafice sau desene. Acest mod de prezentare grafică a rezultatelor este de cele mai multe ori mai sugestiv decît un șir de valori numerice.

5.4.1. CONECTAREA CONVERTOARELOR NUMERIC-ANALOGICE LA MICROPROCESOR

Interfața înregistratorului în coordonate (fig. 5.15) are ca elemente principale două convertoare numeric-analogice formînd un sistem de distribuție de date cu demultiplexare numerică. Aceste convertoare transformă datele numerice de la ieșirile microsistemului în mărimi analogice, tensiuni electrice, care sînt acceptate de către înregistrator ca mărimi de intrare.

Convertoarele integrate, de 8 biți, sînt de tipul *DATEL MC8BC* și au structura prezentată în fig. 5.16. Sînt convertoare multifuncționale putînd fi utilizate ca și convertoare numeric-analogice sau ca și convertoare analog-numerice. În interfața din fig. 5.15 ele realizează conversia numeric-analogică

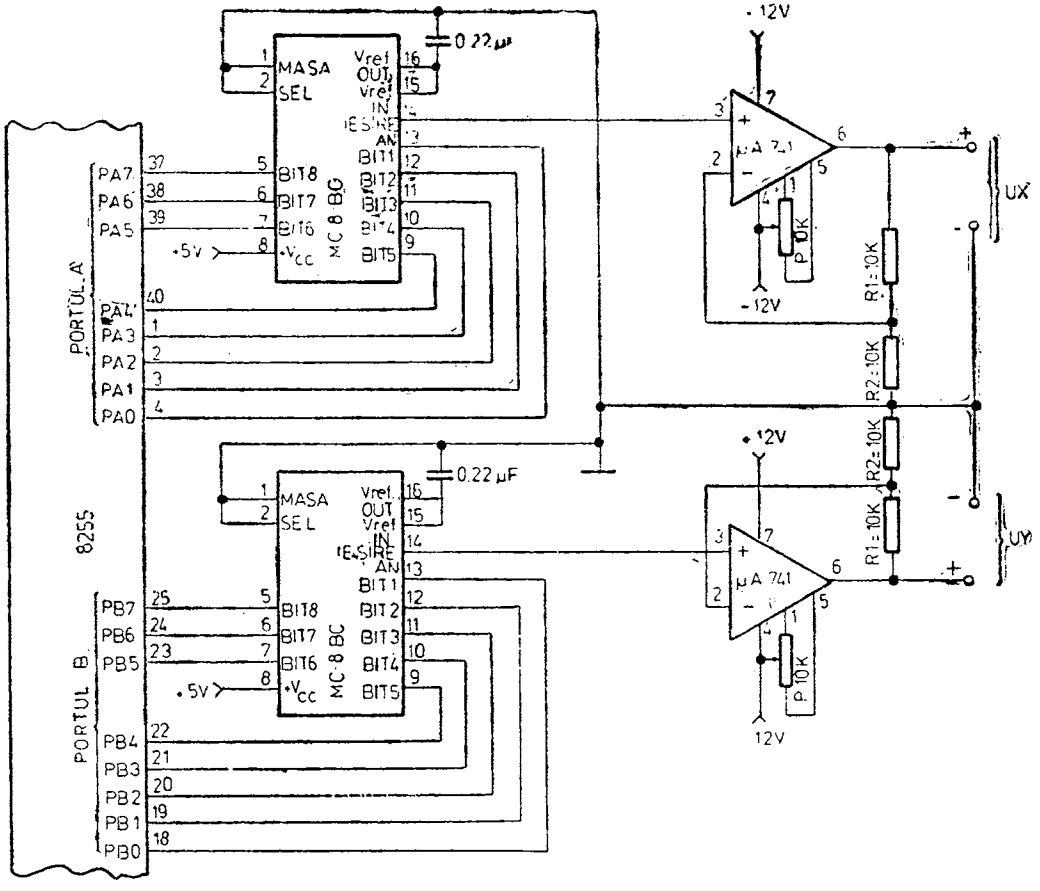


Fig. 5.15. Conectarea convertoarelor numeric-analogice la microsistem

unipolară. La ieșirea analogică a convertorului se obține o tensiune pozitivă, direct proporțională cu valoarea numărului de 8 biți aplicat la intrările comutatoarelor rețelei $R-2R$. În tabelul 5.1 este descrisă această corespondență dintre intrarea numerică și ieșirea analogică a convertorului numeric-analogic operînd în mod unipolar.

Convertorul *MC8BC* necesită o singură tensiune de alimentare ($+5\text{V} \pm 10\,\%$). El are încorporată o sursă internă de referință de $+2{,}55$ V. Capacitatea de 0,22 μF conectată între pinii 1 și 16 asigură stabilizarea acestei surse de referință.

Timpul de stabilizare a tensiunii de la ieșirea convertorului este de 2 μs.

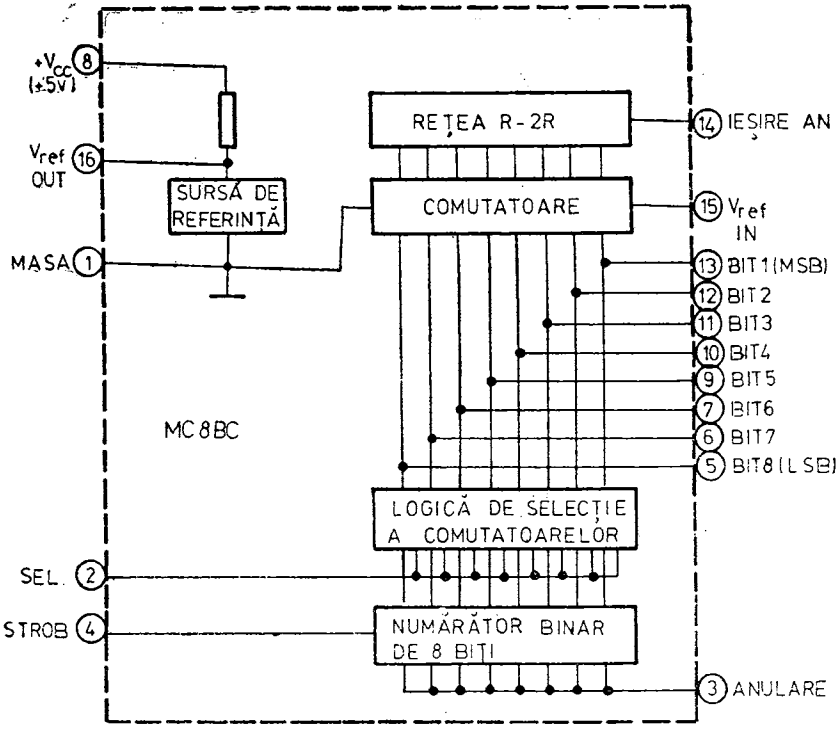


Fig. 5.16. Structura convertorului numeric-analogic/analog-numerice MC 8BC

Folosindu-se tensiunea de referință internă, la ieșirea convertorului numeric analogic se vor obține tensiuni între 0 V și +2,55 V.

La ieșirea fiecărui convertor este conectat cîte un amplificator neinversor realizat pe baza amplificatorului operațional μA 741. Cîștigul amplificatorului $A=1+R1/R2$ este funcție de raportul celor două rezistențe $R1$ și $R2$ astfel încît prin intermediul acestui raport se poate calibra tensiunea analogică de la ieșirea amplificatorului în domeniul (0...+10) V. Pentru cazul considerat în fig. 5.15. cele două rezistența sînt egale, astfel încît $A=2$, ceea ce determină un domeniu de (0...+5,12) V pentru fiecare dintre cele două tensiuni UX și UY.

Amplificatoarele conectate la ieșirile convertoarelor numeric-analogice asigură de asemenea o rezistență de ieșire scăzută a montajului.

Tabelul 5.1.

Intrarea numerică								Ieşirea analogică
DB7	*DB6*	*DB5*	*DB4*	*DB3*	*DB2*	*DB1*	*DB0*	
0	0	0	0	0	0	0	0	$V_{ref} \cdot \frac{0}{256} = 0$
0	0	0	0	0	0	0	0	$V_{ref} \cdot \frac{1}{256}$
0	0	0	0	0	0	0	0	$V_{ref} \cdot \frac{2}{256}$
.								
.								
.								
X	*X*	*X*	*X*	*X*	*X*	*X*	*X*	$V_{ref} \cdot \frac{\sum_{i=0}^{7} DB_i \cdot 2^i}{256}$
.								
.								
.								
1	1	1	1	1	1	1	0	$V_{ref} \cdot \frac{254}{256}$
1	1	1	1	1	1	1	1	$V_{ref} \cdot \frac{255}{256}$

Cele două convertoare numeric-analogice sînt conectate la „portul" *A* şi respectiv „portul" *B* al unui circuit de interfaţă programabil 8255. Acest circuit este astfel programat încît să lucreze în modul cu „porturile" *A* şi *B* operînd ca „porturi" de ieşire.

Pentru a se putea comanda poziţia peniţei înregistratorului s-a conectat microreleul *RELPEN* la primul *PC0* al „portului" *C* al aceluiaşi circuit de interfaţă programabil 8255 (fig. 5.17). Conectarea s-a făcut prin intermediul unui montaj inversor format din rezistenţele *R*3, *R*4 şi *R*5 şi tranzistorul *T*1.

În stare de repaus la *PC0* este un nivel logic *LØ* care menţine tranzistorul *T*1 blocat astfel încît releul este neacţionat şi deci contactul său RELPEN este deschis.

Cînd la *PC*0 apare un nivel logic *HI* baza tranzistorului *T*1 este polarizată de la +5 V prin *R*3 şi *R*4 astfel încît tranzistorul intră în conducţie. În consecinţă releul este acţionat şi contactul său *RELPEM* se închide, comandînd coborîrea peniţei.

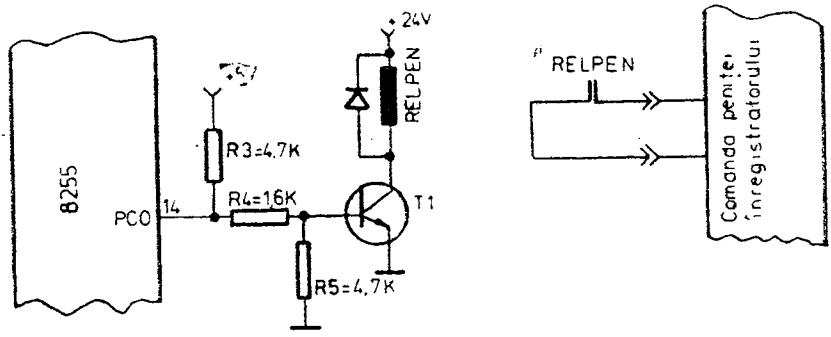


Fig. 5.17. Conectarea microreleului de comandă a poziției peniței înregistratorului

5.4.2. SUBPROGRAMUL PENTRU TRASAREA DE GRAFICE CU AJUTORUL ÎNREGISTRATORULUI ÎN COORDONATE

Subprogramul denumit *GRAFIC*1, furnizează la ieșirile „porturilor" *A* și *B* ale circuitului de interfață programabil 8255, valorile curente, *X* și respectiv *Y*, ale coordonatelor punctului curent al graficului. Subprogramul asigură coborîrea și ridicarea peniței înregistratorului la începutul și respectiv terminarea trasării graficului.

Parametrii de intrare ai subprogramului sînt : *DX* — mărimea pasului pe abscisă, *ADYO* — adresa de început a blocului de valori a ordonatelor și *NP* — numărul de puncte ale graficului. Se presupune că valorile ordonatelor sînt cuvinte de cîte un octet, dispuse în memorie la adrese succesive începînd de la adresa *ADYO* pînă la adresa (*ADY*0+*NP*−1).

Transmiterea parametrilor spre subprogram se face prin încărcarea lor în registrele de lucru ale microprocesorului înainte de apelarea subprogramului : DY se încarcă în registrul *B*, *ADY*0 în registrele (*H*, *L*) iar *NP* se încarcă în registrul *C*.

Subprogramul face, la rîndul său, apel la un subprogram *INTIRZ* care asigură pauza de timp necesară pentru manevrarea peniței înregistratorului și pentru deplasarea cursorului acestuia între punctele succesive ale graficului. Subprogramul *INTIRZ* presupune încărcarea valorii pauzei dorite în registrele (*D*, *E*) înainte de apelarea sa.

Subprogramul GRAFIC 1 afectează toate registrele și toți indicatorii microprocesorului.

Deoarece subprogramul are o structură simplă s-a considerat că nu mai este necesar să se prezinte organigrama sa. Comentariile care însoțesc textul subprogramului scris în limbaj de asamblare sînt suficiente pentru înțelegerea sa.

```
GRAFIC 1 : MVI A, 88H      ;|  Se programează circuitul 8255 în
           ØUT MØDPØRT     ;|  modul 0 cu PA, PB și PC0-PC3 ca
                           ;|  porturi de ieșire iar PC4-PC7 ca
                           ;|  port de intrare.
           MVI A, 00H      ;|  Se poziționează cursorul înregistra-
           MØV D, A        ;|  torului în punctul inițial (0, Y 0) al
           ØUT PØRTA       ;|  graficului.
           MØV A, M        ;|
           ØUT PØRTB       ;|
           PUSH D          ;   Se salvează conținutul registrului D
                               reprezentînd valoarea curentă a
                               abscisei.
           LXI D, T1       ;|  Se asigură pauza T1 pentru a da
                           ;|  timp cursorului să ajungă în punc-
           CALL INTIRZ     ;|  tul inițial al graficului.
           MVI A, 01H      ;|  Se dă comanda de coborîre a peniței
           ØUT PØRTC       ;|  înregistratorului.
           LXI D, T2       ;|  Se asigură pauza T2 pentru termi-
           CALL INTIRZ     ;|  narea coborîrii peniței.
BUCLA :    DCR C           ;|  Se decrementează contorul de punte
                           ;|  rămase din grafic ; în caz de ter-
           JZ STØP         ;|  minare a punctelor se sare la pro-
                           ;|  cedura de terminare a subprogra-
                           ;|  mului.
           PØP D           ;   Se restaurează în D ultima valoare
                               a lui X.
           MØV A, D        ;|  Se calculează noua valoare a absci-
           ADD B           ;|  sei X=X+DX.
           MØV D, A        ;   Se introduce în D noua valoare a
                               abscisei.
           ØUT PØRTA       ;   Se aplică înregistratorului noua va-
                               loare a abscisei.
           INX H           ;   Se pregătește adresa noii ordonate.
           MØV A, M        ;|  Se aplică înregistratorului noua va-
           ØUT PØRT B      ;|  loare a ordonatei.
           PUSH D          ;   Se salvează valoarea abscisei curente
                               conținută în registrul D.
```

```
                LXI  D, T1      ;|  Se asigură pauza T1 necesară pen-
                                 |  tru ajungerea cursorului în noua
                CALL INTIRZ     ;|  poziţie.
                JMP  BUCLA      ;   Se reia ciclul, pentru următorul
                                    punct al graficului.
STØP:           MVI  A, 00H     ;|  Se comandă ridicarea peniţei în-
                ØUT  PØRTC      ;|  registratorului.
                LXI  D, T2      ;|  Se asigură intervalul de timp T2
                CALL INTIRZ     ;|  pentru ridicarea peniţei.
                RET
```

Datorită caracterului secvenţial al programului apare un anumit decalaj între momentele de reactualizare a abscisei şi ordonatei punctului curent al graficului. Acest defazaj este însă practic insesizabil deoarece înregistratorul are o constantă de timp relativă mult mai mare.

Subprogramul de întîrziere are următoarea structură :

```
INTIRZ:         DCX  D          ;   Se decrementează numărul conţinut
                                    în perechea de registre (D, E).
                MØV  A, E       ;|  Se testează ajungerea la zero a
                ØRA  D          |   perechii de registre (D, E).
                RZ              ;   În caz de ajungere la zero se revine
                                    în programul apelant.
                JMP  INTIRZ     ;   Deoarece nu s-a ajuns la zero, se
                                    reia decrementarea.
```

5.4.3. SCRIEREA CARACTERELOR ALFANUMERICE CU AJUTORUL ÎNREGISTRATORULUI ÎN COORDONATE

Posibilitatea de a scrie caractere alfanumerice la înregistratorul în coordonate permite utilizatorului să însoţească graficele cu eventuale comentarii. În acest scop a fost conceput subprogramul *CARALFAN*. Prin apelarea acestui subprogram se poate scrie la înregistratorul în coordonate orice literă majusculă din alfabetul latin sau oricare cifră zecimală.

Scrierea caracterelor alfanumerice se face prin sinteza lor în cadrul unui format patratic cu „9 puncte", aşa cum rezultă din fig. 5.18.

Subprogramul *CARALFAN* furnizează la ieşirile porturilor *A* şi *B* ale interfeţei programabile 8255, succesiunea de perechi de valori *X*, *Y* ale coordonatelor punctelor între care trebuie să se deplaseze peniţa înregistratorului pentru a sintetiza caracterul alfanumeric dorit.

Subprogramul asigură de asemenea ridicarea peniţei înregistratorului pe durata deplasării acesteia din ultima poziţie avută pînă în punctul din punctul

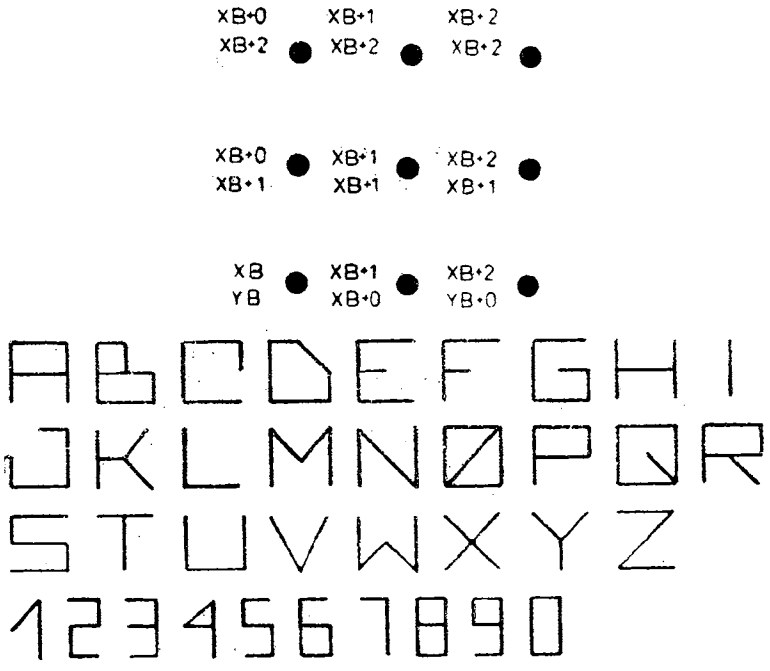


Fig. 5.18. Sinteza caracterelor alfanumerice

din care se începe trasarea caracterului dorit. Peniţa este coborîtă în punctul iniţial al caracterului şi rămîne aşa pe toată durata trasării caracterului. (Pentru a se evita ridicarea şi coborîrea peniţei în timpul trasării caracterului s-a acceptat dublarea unor trasee).

Parametrii de intrare ai subprogramului sînt : *XB* şi *YB* — coordonatele absolute ale punctului de bază (fig. 5.18) al caracterului şi *CØDCAR* — codul caracterului. Parametrii *XB* şi *YB* au lungimea de cîte un octet iar parametrul *CØDCAR* are lungimea de doi octeţi.

Transmiterea parametrilor spre program se face prin încărcarea lor în registrele de lucru ale microprocesorului înainte de apelarea subprogramului. Parametrul *XB* se introduce în registrul *B* iar parametrul *YB* se introduce în *C*. Parametrul *CØDCAR* se introduce în perechea de registre *H, L*.

Subprogramul utilizează subprogramul *INTIRZ* (descris anterior) pentru a asigura pauzele necesare pentru manevrarea căruciorului şi peniţei înregistratorului. Mai sînt utilizate încă două subprograme : *DEPL* — pentru deplasarea peniţei în punctul avînd perechea de coordonate prezentă în acumulator şi *PENIŢA* — pentru ridicarea sau coborîrea peniţei conform ordinului pregătit în acumulator.

În cadrul subprogramului *CARALFAN* se afectează acumulatorul și indicatorii microprocesorului.

Codul caracterului care se trasează — *CØDCAR* — reprezintă de fapt adresa de început a zonei de memorie în care se află succesiunea coordonatelor relative a punctelor prin care se trasează caracterul respectiv. Aceste coordonate sînt relative la punctul de bază (*XB, YB*) considerat întotdeauna ca fiind punctul din stînga-jos în cadrul sistemului de 9 puncte adoptat pentru sinteza caracterelor. Coordonatele relative sînt cotate conform convenției prezentate în fig. 5.18.

Coordonatele relative ale fiecărui punct sînt compactate într-un singur octet. Jumătatea cea mai semnificativă a acestui octet reprezintă coordonata relativă pe axa *X* iar jumătatea cea mai puțin semnificativă coordonata relativă pe axa *Y*. Așa cum rezultă din fig. 5.18 coordonatele relative ale oricăruia dintre cele 9 puncte pot lua cel mult valoarea 2. Ținîndu-se seama de acest aspect s-a stabilit convenția de a marca ultimul punct al oricărui caracter prin punerea unui 1 în poziția de bit cel mai semnificativ al octetului ce reprezintă coordonatele acelui punct. Convenția face astfel posibilă determinarea momentului de sfîrșit a trasării caracterului.

În continuare se prezintă organizarea zonelor de memorie cuprinzînd coordonatele relative ale celor 36 caractere alfanumerice conform fig. 5.18:

Adresa de memorie	Conținutul adresei	Observație
CØDA	00	litera „A"
CØDA+1	01	
CØDA+2	21	
CØDA+3	22	
CØDA+4	02	
CØDA+5	01	
CØDA+6	21	
CØDA+7	A0	
CØDB	00	litera „*B*"
CØDB+1	02	
CØDE+2	12	
CØDE+3	11	
CØDB+4	01	
CØDB+5	21	
CØDB+6	20	
CØDB+7	80	
CØDC	21	litera „*C*"
CØDC+1	22	
CØDC+2	02	
CØDE	22	litera „*E*"
CØDE+1	02	
CØDE+2	01	
CØDE+3	11	
CØDE+4	01	
CØDE+5	00	
CØDE+6	A0	
CØDF	22	litera „*F*"
CØDF+1	02	
CØDF+2	01	
CØDF+3	11	
CØDF+4	01	
CØDF+5	80	
CØDG	22	litera „*G*"
CØDG+1	02	
CØDG+2	00	
CØDG+3	20	
CØDG+4	21	
CØDG+5	91	

Adresa de memorie	Conținutul adresei	Observații
CØDC+3	00	
CØDC+4	A0	
CØDD	00	litera „*D*“
CØDD+1	02	
CØDD+2	12	
CØDD+3	21	
CØDD+4	20	
CØDD+5	80	
CØDJ	12	litera „*J*“
CØDJ+1	22	
CØDJ+2	20	
CØDJ+3	00	
CØDJ+4	81	
CØDK	00	litera „*K*“
CØDK+1	02	
CØDK+2	01	
CØDK+3	11	
CØDK+4	22	
CØDK+5	11	
CØDK+6	A0	
CØDL	02	litera „*L*“
CØDL+1	00	
CØDL+2	A0	
CØDM	00	litera „*M*“
CØDM+1	02	
CØDM+2	11	
CØDM+3	22	
CØDM+4	A0	
CØDN	00	litera „*N*“
CØDN+1	02	
CØDN+2	20	
CØDN+3	A2	
CØDØ	00	litera „*Ø*“
CØDØ+1	22	
CØDØ+2	02	
CØDØ+3	00	
CØDØ+4	20	
CØDØ+5	A2	

Adresa de memorie	Conținutul adresei	Observații
CØDH	00	litera „*H*“
CØDH+1	02	
CØDH+2	01	
CØDH+3	21	
CØDH+4	22	
CØDH+5	A0	
CØDI	10	litera „*I*“
CØDI+1	92	
CØDS	00	litera „*S*“
CØDS+1	20	
CØDS+2	21	
CØDS+3	01	
CØDS+4	02	
CØDS+5	A2	
CØDT	10	litera „*T*“
CØDT+1	12	
CØDT+2	02	
CØDT+3	A2	
CØDU	02	litera „*U*“
CØDU+1	00	
CØDU+2	20	
CØDU+3	A2	
CØDV	02	litera „*V*“
CØDV+1	10	
CØDV+2	A2	
CØDW	02	litera „*W*“
CØDW+1	00	
CØDW+2	11	
CØDW+3	20	
CØDW+4	A2	
CØDX	02	litera „*X*“
CØDX+1	20	
CØDX+2	11	
CØDX+3	00	
CØDX+4	A2	
CØDY	02	litera „*Y*“
CØDY+1	11	
CØDY+2	10	

Adresa de memorie	Conținutul adresei	Observații
CØDP	00	litera „*P*“
CØDP+1	02	
CØDP+2	22	
CØDP+3	21	
CØDP+4	81	
CØDQ	11	litera „*Q*“
CØDQ+1	20	
CØDQ+2	00	
CØDQ+3	02	
CØDQ+4	22	
CØDQ+5	A0	
CØDR	00	litera „*R*“
CØDR+1	02	
CØDR+2	22	
CØDR+3	21	
CØDR+4	01	
CØDR+5	11	
CØDR+6	A0	
*CØD*4	10	cifra „4“
*CØD*4+1	12	
*CØD*4+2	01	
*CØD*4+3	91	
*CØD*5	12	cifra „5“
*CØD*5+1	02	
*CØD*5+2	01	
*CØD*5+3	11	
*CØD*5+4	10	
*CØD*5+5	80	
*CØD*6	12	cifra „6“
*CØD*6+1	02	
*CØD*6+2	00	
*CØD*6+3	10	
*CØD*6+4	11	
*CØD*6+5	81	
*CØD*7	02	cifra „7“
*CØD*7+1	12	
*CØD*7+2	90	
CØDY+3	11	
CØDY+4	A2	
CØDZ	02	litera „*Z*“
CØDZ+1	22	
CØDZ+2	00	
CØDZ+3	AO	
*CØD*1	01	cifra „*1*“
*CØD*1+1	12	
*CØD*1+2	90	
*CØD*2	02	cifra „*2*“
*CØD*2+1	12	
*CØD*2+2	11	
*CØD*2+3	01	
*CØD*2+4	00	
*CØD*2+5	90	
*CØD*3	02	cifra „3“
*CØD*3+1	12	
*CØD*3+2	11	
*CØD*3+3	01	
*CØD*3+4	11	
*CØD*3+5	10	
*CØD*3+6	80	
*CØD*8	11	cifra „8“
*CØD*8+1	01	
*CØD*8+2	02	
*CØD*8+3	12	
*CØD*8+4	10	
*CØD*8+5	00	
*CØD*8+6	81	
*CØD*9	11	cifra „9“
*CØD*9+1	01	
*CØD*9+2	02	
*CØD*9+3	12	
*CØD*9+4	10	
*CØD*9+5	80	
*CØD*0	00	cifra „0“
*CØD*0+1	10	

CØD0+2 12
CØD0+3 02
CØD0+4 80

Subprogramul are următoarea descriere în limbaj de asamblare :

```
CARALFAN:  PUSH  H          ;  Se salvează registrele H şi L.
           PUSH  D          ;  Se salvează registrele D şi E.
           PUSH  B          ;  Se salvează registrele B şi C.
           MVI   A, 88 H    ;  Se programează circuitul 8255 în
                            ;  modul 0 astfel ca PA, PB şi
                            ;  PC0-PC3 să fie porturi de ieşire
           QUT   MØDPØRT    ;  iar PC4-PC7 să fie port de intrare.
           MVI   A, 00H     ;  Se pregăteşte ordinul de ridicare a
                            ;  peniţei.
           CALL  PENIŢA     ;  Se apelează subprogramul de mane-
                            ;  vrare a peniţei conform ordinului
                            ;  pregătit în acumulator.
           MØV   A, M       ;  Se aduce în acumulator conţinutul
                            ;  adresei de memorie CØDCAR (adresă
                            ;  aflată în H şi L în momentul ape-
                            ;  lării subprogramului CARALFAN).
           CALL  DEPL       ;  Se apelează subprogramul care asi-
                            ;  gură deplasarea peniţei în punctul
                            ;  avînd coordonatele relative în A.
           MVI,  A, 01H     ;  Se pregăteşte ordinul de coborîre a
                            ;  peniţei.
           CALL  PENIŢA     ;  Se apelează subprogramul de memo-
                            ;  rare a peniţei.
CICLU:     INX   H          ;  Se pregăteşte adresa următoarei pe-
                            ;  rechi de coordonate a caracterului.
           MØV   A, M       ;  Se aduce în acumulator următoarea
                            ;  pereche de coordonate.
           RAL              ;  Se testează bitul cel mai semnifi-
                            ;  cativ al octetului ce reprezintă pe-
                            ;  rechea de coordonate a punctului
                            ;  următor. În caz că este ultimul
           JC    CNØU       ;  punct al caracterului se termină
           CALL  DEPL       ;  trasarea caracterului.
                            ;  Se apelează subprogramul de deplasare a peniţei în următorul punct
                            ;  (avînd coordonatele în A).
           JMP   CICLU      ;  Se reia ciclul pentru următoarea
                            ;  pereche de coordonate.
```

```
CNØU:   RAR              ;  Se reface conținutul inițial al acu-
                            mulatorului.
        CALL  DEPL       ;  Se apelează subprogramul de deplasare a peniței în vederea deplasării ei în ultimul punct al caracterului.
        PØP   B          ;  Se reface conținutul registrelor B și C.
        PØP   D          ;  Se reface conținutul registrelor D și E.
        PØP   H          ;  Se reface conținutul registrelor H și L.
        RET              ;  Se revine la subprogramul apelant.
```

Subprogramul *DEPL* asigură deplasarea peniței în punctul avînd perechea de coordonate prezentă în acumulator sub forma unui singur octet. Subprogramul dezmembrează octetul în valorile corespunzătoare coordonatei relative pe axa *X* și respectiv coordonatei relative pe axa *Y* a punctului. Subprogramul calculează apoi coordonatele absolute ale punctului pe baza acestor coordonate relative și a coordonatelor absolute de bază *XB* și *YB* (aflate în registrele *B* și respectiv *C*). Subprogramul are următoarea formă :

```
DEPL:   PUSH  PSW        ;  Se salvează acumulatorul și indicatorii.
        PUSH  H          ;  Se salvează registrele H și L.
        PUSH  D          ;  Se salvează registrele D și E.
        PUSH  B          ;  Se salvează registrele B și C.
        ANI   30H        ;  Se extrage prin mascare coordonata relativă pe axa X.
        RRC              ;  Prin rotiri succesive spre dreapta se
        RRC              ;  aduce coordonata relativă pe axa X
        RRC              }  din tetrada cea mai semnificativă în
        RRC              ;  tetrada cea mai puțin semnificativă
                            a octetului.
        ADD   B          ;  Se calculează coodonata absolută pe axa X.
        ØUT   PØRTA      ;  Se reactualizează valoarea coordonatei absolute pe axa X.
        PØP   PSW        ;  Se reface conținutul inițial al acumulatorului și al indicatorilor.
        ANI   03H        ;  Se extrage prin mascare coordonata relativă pe axa Y.
        ADD   C          ;  Se calculează coordonata absolută pe axa Y.
```

```
        ØUT   PØRTB     ;   Se reactualizează valoarea coordo-
                            natei absolute pe axa Y.
        LXI   D, T1     ;\  Se asigură pauza T1 necesară pen-
        CALL  INTIRZ    ;/  tru deplasarea peniței în noul punct.
        PØP   B         ;   Se reface conținutul registrelor B și C
        PØP   D         ;   Se reface conținutul registrelor D
                            și E.
        PØP   H         ;   Se reface conținutul registrelor H
                            și L.
        RET             ;   Se revine la paragraful apelant.
```

Subprogramul *PENIȚA* asigură manevrarea peniței înregistratorului conform ordinului de coborîre sau ridicare aflat în acumulator în momentul apelării subprogramului. Subprogramul are următoarea descriere în limbaj de asamblare :

```
PENIȚA: ØUT   PØRTC     ;   Se plasează în portul C al circui-
                            tului 8255 ordinul existent în acu-
                            mulator.
        PUSH  D         ;\  Se asigură pauza T2 necesară pentru
        LXI   D, T2     ;|  executarea manevrei de ridicare sau
        CALL  INTIRZ    ;|  coborîre a peniței.
        PØP   D         ;/
        RET             ;   Se revine în programul apelant.
```

În continuare se prezintă un exemplu de utilizare a subprogramului *CARALFAN* pentru scrierea unui text la înregistratorul în coordonate.

Deoarece convertoarele numeric-analogice pentru cele două coordonate sînt de 8 biți rezultă că penița înregistratorului se poate mișca într-un cîmp plan de 256×256. Cuanta de deplasare pe oricare axă este 1. În cadrul acestui cîmp un caracter ocupă o zonă patrată de 2+2. În program se consideră un spațiu minim de 1 pentru pauza între două litere consecutive sau între două rînduri de caractere.

Pentru a se trasa pe două rînduri textul : *MICRØPRØCESØR* 8080 avînd pe (0, 200) și (12,197) drept perechi de coordonate absolute de bază ale primelor caractere de pe cele două rînduri, se folosește următoarea procedură :

— începînd de la adresa *LINIEI* se dispun într-un tabel în memorie, una după alta, adresele de început ale zonelor care conțin coordonatele relative ale caracterelor care trebuiesc trasate în prima linie. După ce s-a înscris codul ultimului caracter din linie, tabelul se încheie prin înscrierea pe ultima poziție a unui octet special *TERMLINIE*, care nu poate fi cod de caracter.

```
LINIE1 :    CØDM        ;   octetul cel mai puțin semnificativ
                            al adresei CØDM.
```

*LINIE*1+01 :	$CØDM_H$	;	octetul cel mai semnificativ al adresei *CØDM*.
*LINIE*1+02 :	$CØDI_L$		
*LINIE*1+03 :	$CØDI_H$		
*LINIE*1+04 :	$CØDC_L$		
*LINIE*1+05 :	$CØDC_H$		
*LINIE*1+06 :	$CØDR_L$		
*LINIE*1+07 :	$CØDR_H$		
*LINIE*1+08 :	$CØDØ_L$		
*LINIE*1+09 :	$CØDØ_H$		
*LINIE*1+0A :	$CØDP_L$		
*LINIE*1+0B :	$CØDP_H$		
*LINIE*1+0C :	$CØDR_L$		
*LINIE*1+0D :	$CØDR_H$		
*LINIE*1+0E :	$CØDØ_L$		
*LINIE*1+0F :	$CØDØ_H$		
*LINIE*1+10 :	$CØDC_L$		
*LINIE*1+11 :	$CØDC_H$		
*LINIE*1+12 :	$CØDE_L$		
*LINIE*1+13 :	$CØDE_H$		
*LINIE*1+14 :	$CØDS_L$		
*LINIE*1+15 :	$CØDS_H$		
*LINIE*1+16 :	$CØDØ_L$		
*LINIE*1+17 :	$CØDØ_H$		
*LINIE*1+18 :	$CØDR_L$		
*LINIE*1+19 :	$CØDR_H$		
*LINIE*1+20 :	*TERMLINIE*;		octetul de marcaj a terminării liniei.

— se organizează un tabel similar conținînd codurile (de cîte 16 biți) ale caracterelor care se doresc a fi trasate pe a doua linie. Fie *LINIE*2 adresa de început a celui de al doilea tabel. Și cel de-al doilea tabel cu coduri de caractere se încheie prin același octet *TERMLINIE* (neconfundabil cu nici un cod de caracter) care marchează terminarea caracterelor din linie.

*LINIE*2	$CØD8_L$	;	Octetul mai puțin semnificativ și octetul cel mai semnificativ al codului caracterului 8.
*LINIE*2+01 :	$CØD8_H$	;	
*LINIE*2+02 :	$CØDO_L$		
*LINIE*2+03 :	$CØDO_H$		
*LINIE*2+04 :	$CØD8_L$		
*LINIE*2+05 :	$CØD8_H$		
*LINIE*2+06 :	$CØDO_L$		
*LINIE*2+07 :	$CØDO_H$		
*LINIE*2+08 :	*TERMLINIE*;		octetul de marcaj a terminării liniei

— se pregăteşte un subprogram (*TIPLINIE*) destinat trasării la înregistrator a unei linii conţinînd caractere descrise printr-un tabel de caractere de tipul celor de mai sus. Subprogramul are următorii parametri: *LINIE* — adresa de început a tabelului de coordonate şi *CØRDBAZA* — coordonatele absolute ale punctului de bază a primului caracter din componenta liniei. Transmiterea parametrilor spre subprogram se face prin încărcarea lor în registrele microprocesorului: *LINIE* se încarcă în perechea de registre (*D,E*) iar *CØRDBAZA* trebuie încărcat în perechea de registre (*B,C*).

```
TIPLINIE: PUSH  H        ;  Se salvează conţinutul registrelor H şi L.
ADR1:     LDAX  D        ;  Octetul mai puţin semnificativ al codului caracterului curent este
                             extras din tabel şi încărcat, prin-
          MØV   L,A      ;  intermediul acumulatorului, în registrul L.
          SUI   TERMLINIE ;  Se testează dacă nu s-a depistat
          JZ    ADR5     ;  octetul de marcaj a terminării liniei.
          INR   E        ;  Se calculează adresa următoare
          JZ    ADR3     ;  din tabelul codurilor de caractere.
ADR2:     LDAX  D        ;  Octetul cel mai semnificativ al codului caracterului curent este
                             extras din tabel şi încărcat, prin
          MØV   H,A      ;  intermediul acumulatorului, în registrul H.
          SUI   TERMLINIE ;  Se testează dacă nu s-a depistat
          JZ    ADR5     ;  octetul de marcaj a terminării liniei.
          CALL  CARALFAN :  Se apelează subprogramul de trasare a caracterului curent (existînd pregătite în B şi C coordonatele absolute ale punctului de
                             bază a acestui caracter iar în H şi L codul caracterului curent).
          INR   B        ;  Se pregăteşte coordonata absolută pe axa X a punctului de bază a
          INR   B        ;  următorului caracter (Se presupune că un caracter are lăţimea
          INR   B        ;  egală cu 2 iar distanţa între caractere este egală cu 1).
```

```
        INR  E          ;  Se calculează următoarea adresă
        JZ   ADR4       ;  din tabelul codurilor de caractere.
        JMP  ADR1       ;  Se reia ciclul pentru următorul
                           caracter.
ADR3:   INR  D          ;  La calcularea adresei din tabelul
                           de coduri a apărut transfer spre
                           octetul cel mai semnificativ al
                           adresei, transfer ce se adună la
        JMP  ADR2       ;  acest octet.
ADR4:   INR  D          ;  La calcularea adresei din tabelul
                           de coduri a apărut transfer spre
                           octetul cel mai semnificativ al
                           adresei, transfer ce se adună la
        JMP  ADR1       ;  acest octet.
ADR5    PØP  H          ;  Se reface conținutul registrelor
                           H și L.
        RET             ;  Se revine în programul apelant.
```

— folosindu-se subprogramul *TIPLINIE* și avînd pregătite tabelele codurilor de caractere pentru fiecare linie a textului dorit, rezultă în final programul principal:

```
        LXI  B, 00C8H   ;  Se încarcă în perechea de registre
                           (B,C) echivalentul hexazecimal al
                           perechii de coordonate absolute a
                           punctului de bază al primului
                           caracter din linia întîi.
        LXI  D, LINIE1  ;  Se încarcă în perechea de registre
                           (D,E) adresa de început a tabe-
                           lului care conține codurile carac-
                           terelor primei linii.
        CALL TIPLINIE   ;  Se apelează subprogramul de tra-
                           sare la înregistrator a unei linii
                           de caractere.
        LXI  B, 12C5H   ;  Se încarcă în registrele (B,C)
                           echivalentul hexazecimal al pe-
                           rechii de coordonate absolute a
                           punctului de bază al primului
                           caracter din linia a doua.
        LXI  D, LINIE2  ;  Se încarcă în registrele (D,E)
                           adresa de început a tabelului care
```

conține codurile caracterelor din a doua linie.

CALL TIPLINIE ; Se apelează subprogramul de trasare a unei linii de caractere.

5.4.4. DIGITALIZAREA CURBELOR DESENATE FOLOSINDU-SE ÎNREGISTRATORUL ÎN COORDONATE X–Y CONECTAT LA MICROSISTEM

Înlocuindu-se peniţa înregistratorului cu un sesizor optic se poate realiza digitalizarea curbelor desenate pe hîrtie.

Sesizorul optic furnizează la ieşire un semnal $\overline{SETS\emptyset}$ (*TTL*-compatibil) avînd valoarea logică 0 în momentul sesizării unei zone întunecate a desenului şi valoarea logică 1 atunci cînd este în dreptul hîrtiei albe, fără desen.

Conectarea sesizorului optic la interfaţa programabilă 8255 al microsistemului se face prin intermediul unui circuit basculant bistabil *R-S* realizat din două porţi *ŞI-NU*. În fig. 5.19 se prezintă acest mod de interfaţare a sesizorului optic cu microsistemul. Circuitul basculant *R-S* are ca rol eliminarea zgomotelor ce apar în semnalul $\overline{SETS\emptyset}$ datorită vibraţiilor căruciorului înregistratorului pe care este montat sesizorul.

Pot fi digitalizate curbe plasate în zona delimitată de valorile 0 şi *UXMAX* de pe axa *X* şi valorile 0 şi *UYMAX* de pe axa *Y*. Mărimile analogice *UXMAX* şi *UYMAX* corespund valorilor numerice *XMAX=FFH* şi respectiv *YMAX= =FFH* furnizate de „porturile" *A* şi *B* ale circuitului programabil 8255. Fineţea de digitalizare este de 8 biţi pentru fiecare dintre cele două axe.

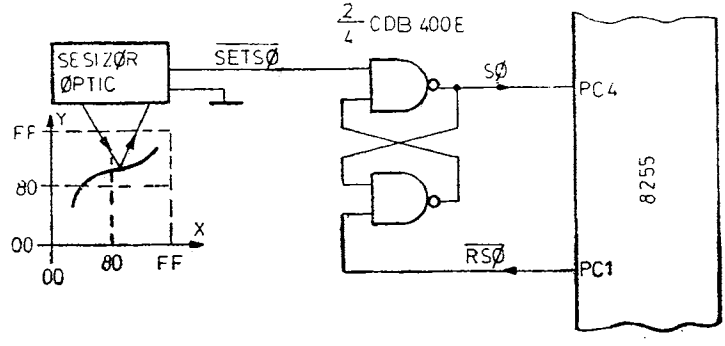


Fig. 5.19. Conectarea sesizorului optic la circuitul programabil 8255 pentru digitalizarea curbelor desenate folosind înregistratorul în coordonate comandat de microprocesor.

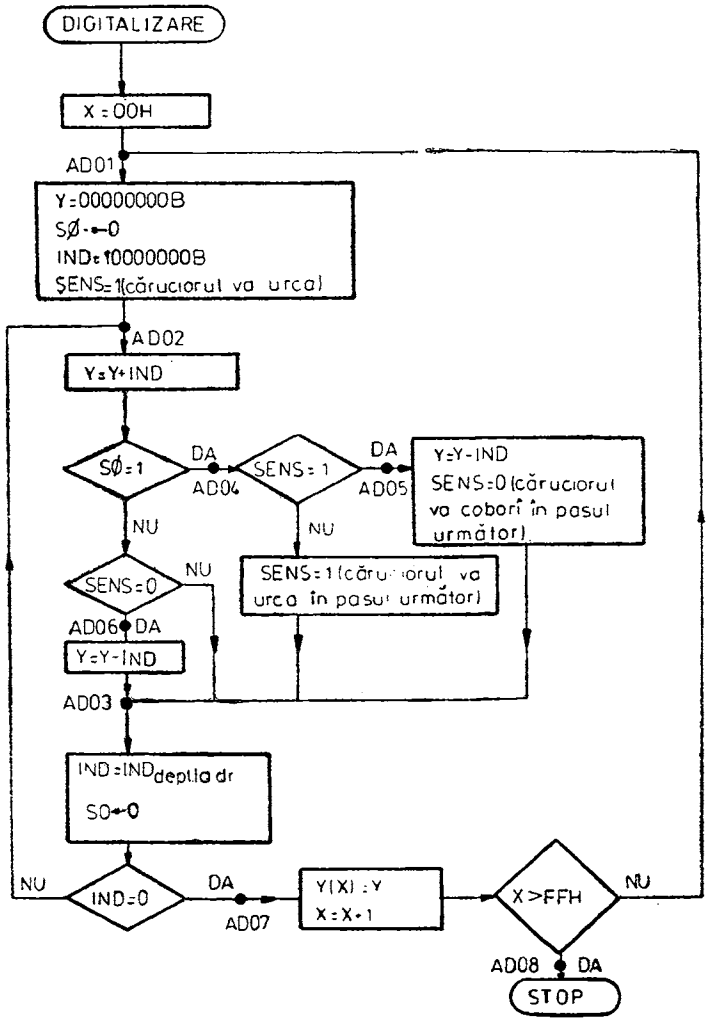


Fig. 5.20. Organigrama procedurii de digitalizare

Digitalizarea se realizează prin deplasarea cu pas constant a căruciorului pe axa X a înregistratorului și determinarea ordonatei Y a curbei desenate, pentru fiecare poziție a căruciorului pe axa X. Determinarea valorii ordonatei se face prin procedeul de înjumătățire, binecunoscut în tehnica construcției

convertoarelor analog-numerice [1]. Folosirea acestui procedeu permite să se determine valoarea oricărei ordonate în maximum 8 paşi.

Procedura de digitalizare este descrisă prin organigrama din fig. 5.20. Prin această procedură se delimitează din ce în ce mai strîns zona pe verticala abscisei unde se găseşte orconata corespunzătoare a curbei.

În primul pas al acestei proceduri de delimitare cursa pe verticală a căruciorului care poartă sesizorul optic este jumătate din cursa sa maximă. Cursa căruciorului se înjumătăţeşte de la pas la pas, ajungînd în final (la al optulea pas) să fie doar o cuantă.

În fiecare pas al delimitării pe verticală se decide valoarea cîte uneia din cei 8 biţi ai ordonatei. În primul pas se decide valoarea bitului cel mai semnificativ, ajungîndu-se la al optulea pas să se decidă şi valoarea bitului cel mai puţin semnificativ. Poziţia bitului 1 în cadrul indicatorului *IND* din organigramă marchează bitul ordonatei care se determină în pasul respectiv.

Valoarea care se atribuie bitului care se determină într-un pas oarecare depinde pe de o parte de răspunsul sesizorului optic în pasul respectiv şi pe de altă parte de sensul de deplasare (ridicare sau coborîre) a căruciorului în acel pas.

Corespunzător acestei proceduri rezultă următorul program scris în limbaj de asamblare 8080 (*X* este memorat în registrul *B*, *Y* în registrul *C*, indicatorul *IND* în rgistrul *D*, indicatorul *SENS* în registrul *E*) :

```
        MVI  A, 88H      ;  Se programează circuitul 8255 în
                            modul 0 cu PA, PB şi PC0÷PC3
                            ca „porturi" de ieşire iar PC4÷PC7
        ØUT  MØDPØRT     ;  ca „port" de intrare.
        MVI  B, 00H      ;  Se iniţializează poziţia căruciorului
        MØV  A,B         ;  cu sesizorul optic pe axa X X=00H;
        ØUT  PØRTA       ;
        LXI  H. ADINREZ  ;  Se iniţializează adresa de început a
                            blocului cu rezultatele.
AD01 :  MVI  A, 00H      ;  S-a generat un impuls negativ pentru
        ØUT  PØRTC       ;  anularea bistabilului SØ;
        MVI  A, 02H      ;
        ØUT  PØRTC       ;
        MVI  C, 00H      ;  Se iniţializează poziţia căruciorului
        MØV  A, C        ;  cu sesizorul optic pe axa Y
        ØUT  PØRTB       ;  Y=00000000B ;
        MVI  D,80H       ;  Se iniţializează indicatorul IND
                            (IND=10000000B) ;
        MVI  E, 01H      ;  Se iniţializează indicatorul SENS
                            (SENS=1) ;
AD02 :  MØV  A, C        ;  Se modifică poziţia căruciorului cu
        ADD  D           ;  sesizorul optic, luîndu-se poziţia
        ØUT  PØRTB       ;  Y=Y+IND ;
```

```
        PUSH  D          ;
        LXI   D, T       ;   Se asigură întîrzierea T necesară
        CALL  INTIRZ     ;   deplasării căruciorului ;
        PØP   D          ;

        IN    PØRTC      ;
        ANI   10H        ;   SØ=1 ?
        JNZ     AD04     ;

        XRA   A          ;
        ADD   E          ;   SENS=0 ?
        JZ    AD06       ;

AD03 :  MØV   A,D        ;
        PRC              ;   IND=IND deșl. la dr.
        MØV   D,A        ;

        MVI   A, 00H     ;
        ØUT   PØRTC      ;
        MVI   A, 02H     ;   SØ=0
        ØUT   PORTC      ;

        XRA   A          ;
        ADD   D          ;   IND=0 ?
        JZ    AD07       ;
        JMP   AD02       ;

AD04 :  XRA   A          ;
        ADD   E          ;   SENS=1 ?
        JNZ   AD05       ;
        MVI   E,01H      ;   SENS=1 ;
        JMP   AD03       ;

AD05 :  MØV   A, C       ;
        SUB   D          ;   Y=Y-IND ;
        MØV   C,A        ;
        MVI   E,00H      ;   SENS=0 ;
        JMP   AD03       ;

AD06 :  MØV   A,C        ;
        SUB   D          ;   Y=Y-IND ;
        MØV   C,A        ;
        JMP   AD03       ;

AD07 :  MØV   M,C        ;   Se memorează ordonata digitalizată
                             Y (X)=Y ;

        INR   L          ;   Se incrementează adresa curentă a
        JZ    AD27       ;   blocului de rezultate.
```

```
AD17 :  INR  B          ;     X=X+1 ;
        MØV  A,B        ;
        SUI  FFH        ;     X>FFH ? ;
        JNC  AD08       ;
        JMP  AD01       ;
AD27 :  INR  H          ;     Se incrementează octetul cel mai
        JMP  AD17       ;     semnificativ al adresei curente a
                              blocului de rezultate ;
AD08 :  HLT             ;
```

Ca rezultat al programului se obţin 256 de valori ale ordonatelor curbei digitalizate, valori memorate într-un bloc de memorie începînd de la adresa *ADINREZ*, şi ţinînd pînă la adresa *ADINREZ+FFH* inclusiv.

BIBLIOGRAFIE

1. E. Popp ; V. Stoica, *„Principii şi metode de măsurare numerică“*, Ed. Facla, Timişoara, 1977.
2. D. H. Sheingold (Editor) : *„Analog-digital conversion notes“*; ANALOG DEVICES Inc.-USA, 1977.
3. M. Sâmpăleanu, *„Circuite pentru conversia datelor“*, Ed. Tehnică, Bucureşti, 1980.
4. *„Data Acquisition Components and subsystems“*; ANALOG DEVICES Inc.-USA, 1977.
5. D. P. Burton, A. L. Dexter : *Microprocessor Systems Handbook*, ANALOG DEVICE Inc.-USA, 1977.

CAPITOLUL 6

IMPLEMENTAREA UNOR FUNCȚIUNI ARITMETICE DE BAZĂ

În cazul aplicațiilor industriale, microprocesorul execută un program fix, dedicat aplicației căreia îi este destinat. Astfel nu mai este necesară implementarea unui sistem de operare ca și în cazul microcalculatoarelor universale. În cadrul acestui program vor fi implementate doar proceduri care sînt necesare pentru rezolvarea problemelor din aplicația respectivă.

Proiectantul unui microsistem dedicat trebuie să aibă în vedere optimizarea ansamblului celor două componente „hardware"- „software" ale sistemului astfel încît acesta să realizeze performanțele cerute la un preț de cost minim. Programele implementate, la fel ca și partea de circuite a sistemului, trebuie să fie concepute cît mai economic atît din punct de vedere al vitezei de prelucrare cît și din punct de vedere al lungimii lor. Pe lîngă procedurile de implementare a sistemelor de „logică programată", proiectanții de microsisteme dedicate aplicațiilor industriale utilizează și diverse proceduri de calcul aritmetic. Dacă literatura referitoare la utilizările n icroprocesoarelor destinată inginerilor electroniști este abundentă din punct de vedere al soluțiilor „hardware", nu același lucru se poate spune despre soluțiile „software", în general, și despre programele matematice, în special. Programele descrise în acest gen de literatură au ca menire în primul rînd să ilustreze modul de folosire a listei de instrucții a microprocesorului [1], [2]. Bibliotecile matematice ale microcalculatoarelor universale produse industrial cunosc și ele o circulație restrînsă, neînsoțind microsistemele livrate sub formă de componente.

Implementarea funcțiilor aritmetice de bază nu mai este așa de simplă în cazul programării în cod mașină (utilizată în marea parte a aplicațiilor dedicate ale microprocesoarelor) cum este în cazul programării într-un limbaj superior. Rezultă astfel importanța pe care o are, pentru proiectantul microsistemelor dedicate, cunoașterea algoritmilor de efectuarea a operațiilor aritmetice într-un calculator folosind instrucțiile aritmetice slabe ale „codului-mașină".

6.1. REPREZENTAREA DATELOR NUMERICE

6.1.1. FORMATE DE REPREZENTARE

După modul de poziționare a virgulei în decursul efectuării calculelor, există două posibilități de a reprezenta datele numerice în interiorul microprocesorului :

— reprezentarea în sistemul cu virgulă fixă ;

— reprezentarea în sistemul cu virgulă mobilă ;

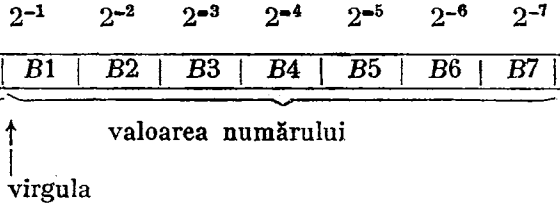
În cadrul acestui capitol se vor face referiri doar la reprezentarea în virgulă fixă. Stabilirea poziției virgulei în sistemul de reprezentare cu virgulă fixă determină domeniul de reprezentare a datelor. Acest lucru nu este însă deosebit de important deoarece virgula fiind fixă se poate folosi un factor de scală ales în funcție de datele unei anumite probleme. După înmulțirea cu factorul de scară datele pot fi aduse în domeniul de reprezentare dorit. Ceea ce este însă cu adevărat important pentru stabilirea poziției virgulei este necesitatea ca rezultatele operațiilor aritmetice să fie și ele cuprinse în domeniul de reprezentare stabilit pentru date [3].

În figurile 6.1 și 6.2 sînt reprezentate graficele probailităților de ieșire din domeniul de reprezentare a rezultatelor operațiilor aritmetice elementare în două cazuri limită de poziționare a virgulei. În ambele cazuri au fost considerați operanzi cu semn avînd 7 biți pentru codificarea valorii numerice a acestora.

În primul caz operanzii sînt numere întregi aparținînd domeniului -127, $+127$. Se poate vedea din fig. 6.1 că în cazul înmulțirii există doar o zonă redusă (nehașurată) a perechilor de operanzi X, Y pentru care rezultatul $(X \cdot Y)$ nu depășește domeniul de reprezentare.

În acest prim caz al operanzilor numere întregi, împărțirea nu duce niciodată la depășiri ale domeniului de reprezentare.

În cel de-al doilea caz operanzii sînt numere subunitare, virgula fiind plasată la extrema stîngă a valorii numerice a acestora, imediat după bitul de semn :


2⁻¹ 2⁻² 2⁻³ 2⁻⁴ 2⁻⁵ 2⁻⁶ 2⁻⁷ ← ponderile biților care
BS B1 B2 B3 B4 B5 B6 B7
reprezintă valoarea
bitul de semn
valoarea numărului
virgula
numărului

În acest caz operanzii aparțin domeniului $-\frac{127}{128}, \frac{127}{128}$. Operația de înmulțire nu mai duce la depășiri ale limitelor extreme ale domeniului de

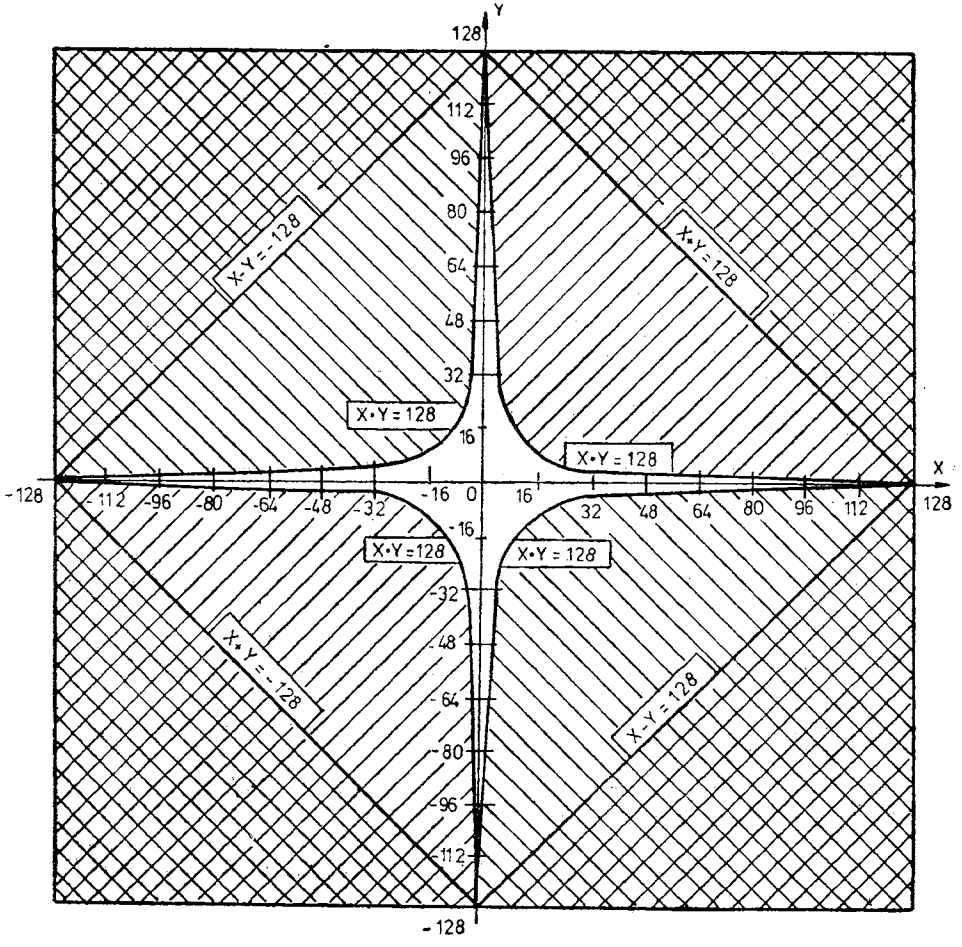


Fig. 6.1. Probabilităţile de depăşire a domeniului de reprezentare de către rezultatul operaţiilor de adunare/scădere şi de înmulţire în cazul reprezentării datelor în virgulă fixă avînd virgula plasată la extremitatea dreaptă (au fost considerate numere întregi din domeniul —127, +127). Zonele de depăşire sînt haşurate.

reprezentare. Înmulţirea poate duce doar la eroriatunci cînd rezultatul este mai mic în valoare absolută decît 1/128. În această situaţie se obţine aşa-numitul „zero de calculator" cu toate că valoarea produsului nu este în realitate nulă. Eroarea care apare, mică în valoare absolută, este de cele mai multe ori acceptabilă.

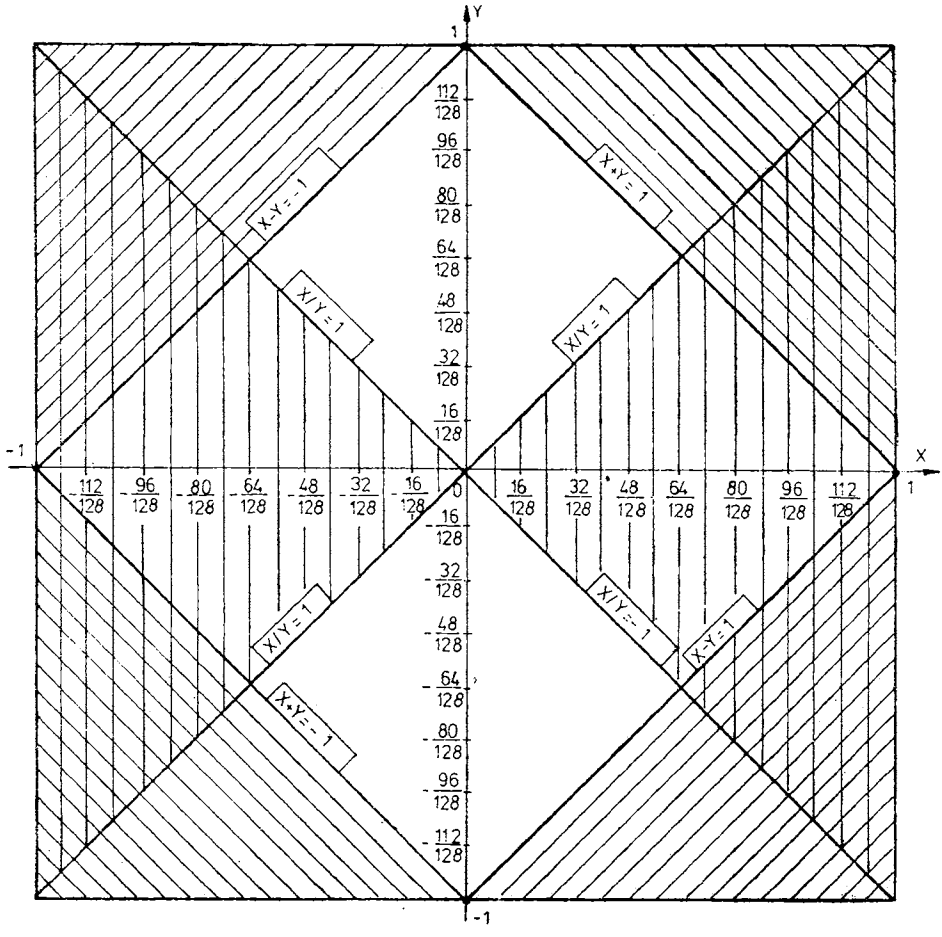


Fig. 6.2. Probabilitatea de depășire a domeniului de reprezentare de către rezultatul operațiilor de adunare/scădere și de împărțire în cazul reprezentării datelor în virgulă fixă avînd virgula plasată la extremitatea stîngă (numere subunitare). Zonele de depășire sînt haşurate.

În cazul reprezentării numerelor sub formă subunitară, operația de împărțire poate duce la depășiri ale limitelor extreme ale domeniului de reprezentare (zona cu haşuri verticale în fig. 6.2).

Comparînd cele două modalități de poziționare a virgulei rezultă clar superioritatea reprezentării operanzilor ca numere subunitare avînd virgula plasată în extremitatea stîngă, imediat după bitul de semn.

6.1.2. CODURI BINARE PENTRU REPREZENTAREA DATELOR

După modul de codificare a numerelor negative există trei tipuri de coduri :
— codul direct (semn-mărime) ;
— codul invers (complementar față de „baza sistemului de numerație —1").
— codul complementar (complementar față de baza sistemului de numerație);
Dintre acestea codul complementar este cel mai des folosit pentru reprezentarea numerelor în decursul prelucrării aritmetice a acestora.

În cazul codului binar complementar (față de 2) un număr pozitiv se reprezintă printr-un cuvînt de cod binar avînd „0" ca bit de semn urmat de ceilalți biți care reprezintă valoarea binară a respectivului număr. Un număr negativ se reprezintă prin inversarea tuturor biților din expresia binară a numărului pozitiv avînd aceeași valoare absolută și adunarea unui 1 la rangul cel mai puțin semnificativ al codului invers (adunarea făcîndu-se cu propagarea transferului).

De exemplu numerele subunitare +67/128 și —67/128 au următoarele reprezentări în codul complementar :

+67/128	0,1000011 ;
—67/128	1,0111100+
	1
	1,0111101;

Folosindu-se codul complementar pot fi reprezentate numere din domeniul [—1, 1). Numărul —1 se codifică în acest caz : 1,0000000. Numărul 0 are în cazul codului complementar o singură reprezentare : 0,0000000.

6.2. IMPLEMENTAREA OPERAȚIILOR ARITMETICE ELEMENTARE ÎNTRE OPERANZI REPREZENTAȚI ÎN COD COMPLEMENTAR FAȚĂ DE 2

6.2.1. SUBPROGRAM DE ADUNARE A DOI OPERANZI CU SEMN DE CÎTE 16 BIȚI, REPREZENTAȚI ÎN COD COMPLEMENTAR FAȚĂ DE 2

Fie $X=XS, X1\ X2 \ldots X15$ și $Y=YS, Y1\ Y2 \ldots Y15$ cei doi operanzi iar $Z=ZS, Z1\ Z2 \ldots Z15$ rezultatul adunării.

În momentul apelării subprogramului, care se face prin instrucția *CALL*, cei doi operanzi trebuie să se afle deja introduși în stivă. Considerînd că (SP_0)

este conținutul registrului *SP* în momentul apelării subprogramului operanzii vor fi dispuși în stivă după cum urmează :

	Adresa de memorie	Conținutul memoriei
Vîrful stivei →	(SP_0)	*Y8* *Y15*
	$(SP_0)+1$	*YS Y1* *Y7*
	$(SP_0)+2$	*X8* *X15*
	$(SP_0)+3$	*XS X1* *X7*

Adunarea $Z=X+Y$ se efectuează conform următorului algoritm :

— se citesc din stivă cei doi operanzi și se introduc în registrele microprocesorului ;

— se efectuează adunarea folosindu-se instrucția de adunare în dublă precizie ;

— se detectează și se semnalizează apariția depășirii capacității registrelor (*DCR*) ca urmare a adunării ;

— se înscrie în stivă rezultatul adunării astfel încît la revenirea din subprogram rezultatul apare în stivă după cum urmează :

	Adresa de memorie	Conținutul memoriei
Vîrful stivei →	(SP_0)	*Z8* *Z15*
	$(SP_0)+1$	*ZS Z1* *Z7*

Apariția sau neapariția *DCR* este indicată de subprogram prin înscrierea octetului *FFH* sau respectiv *00H* la o anumită adresă *ADR DCR* rezervată în memorie pentru acest scop.

În continuare se prezintă subprogramul *DADC2* scris în limbaj de asamblare 8080 :

```
DADC2:  INX  SP          ;\  (SP0)→SP;
        INX  SP          ;/
        LXI  H, ADRDCR   ;\  0→M<ADRDCR>; inițializarea
        MVI  M, 00H      ;>  locației de memorie destinată sem-
                         ;/  nalizării DCR;
        PØP  D               [YS, Y1...Y7]→D, [Y8...Y15]
                             →E și (SP0)+2→SP;
        PØP  H           ;   [XS, X1 ... X7]→ H, [X8 ...
                             ... X15]→L și (SP0)+4→SP;
        MØV  B, H        ;   [XS, X1 ... X7]→B; se salvează
                             octetul care conține semnul primului
                             operand.
        DAD  D           ;   Z (=X+Y)→[H, L]; [ZS, Z1 ...
                             ... Z7]→H și [Z8...Z15]→L;
```

```
                MØV A, B        ;|  XS⊕ZS→CY;
                XRA H           ;|
                RLC             ;|
                JC PØSDCR       ;   Dacă XS⊕ZS=1 este posibil DCR;
CONTINUA:       DCX SP          ;|  (SP0)+2→SP;
                DCX SP          ;|
                PUSH H          ;   [ZS, Z1 ... Z7]→M<(SP0)+1>
                                    și [Z8 ... Z15]→M <(SP0)>,
                                    (SP0)→SP.
                DCX SP          ;|  (SP0)−2→SP;
                DCX SP          ;|
                RET             ;   Revenirea din subprogram;
PØS DCR:        MØV A, D        ;|
                XRA H           ;|  YS⊕ZS→CY;
                RLC             ;|
                JNC CØNTINUA    ;   Dacă YS⊕ZS=0 nu este DCR
                                    chiar dacă s-a găsit că XS⊕ZS=1;
                XTHL            ;   Se salvează în stivă conținutul lui
                                    [H, L];
                LXI H, ADRDCR   ;|  FFH→M<ADRDCR>; se mar-
                MVI M, FFH      ;|  chează apariția DCR înscriindu-se
                                    FFH în locația de memorie pre-
                                    stabilită;
                XTHL            ;   Se readuce din stivă conținutul ini-
                                    țial al lui [H, L];
                JMP CØNTINUA    ;   Se continuă procedura de încheiere
                                    a subprogramului.
```

6.2.2. SUBPROGRAM DE SCĂDERE A DOI OPERANZI CU SEMN, DE CÎTE 16 BIȚI, REPREZENTAȚI ÎN COD COMPLEMENTAR FAȚĂ DE 2

Fie, ca și în cazul precedent, X și Y cei doi operanzi iar $Z=X-Y$ rezultatul scăderii.

Se presupune că în momentul apelării subprogramului, printr-o instrucție ***CALL***, cei doi operanzi trebuie să se afle deja introduși în stivă, după cum urmează:

Vîrful stivei

	Adresa de memorie	Conținutul memoriei
⟵	(SP_0)	$Y8$ ..... $Y15$
	$(SP_0)+1$	XS $Y1$ ..... $Y7$
	$(SP_0)+2$	$X8$ ..... $X15$
	$(SP_0)+3$	XS $X1$ ..... $X7$

Scăderea $Z=X-Y$ se efectuează conform următorului algoritm :
— se citesc din stivă cei doi operanzi și se introduc în registrele microprocesorului;
— se efectuează scăderea octet cu octet a celor doi operanzi ;
— se detectează și se semnalizează apariția *DCR ;*
— se înscrie în stivă rezultatul acesta fiind prezentat astfel :

	Adresa de memorie	Conținutul memoriei
Vîrful stivei →	(SP_0)	$Z8$ ..... $Z15$
	$(SP_0)+1$	ZS $Z1$ ..... $Z7$

Apariția sau neapariția *DCR* este indicată de subprogram prin înscrierea octetului *FFH* sau respectiv *00H* la adresa *ADRDCR* prestabilită în memorie pentru indicarea *DCR.*

În continuare se prezintă subprogramul *DSCC2* (pentru efectuarea scăderii) scris în limbaj de asamblare 8080 :

```
DSCC2 :   INX  SP          ;  (SP0)→SP ;
          INX  SP          ;
          LXI  H, ADRDCR   ;  0→M<ADRDCR> ;
          MVI  M, 00H      ;
          PØP  D           ;  Y→[D, E] ;
          PØP  H           ;  X→[H, L] ;
          MØV  B, H        ;  [XS, X1 ... X7]→B ; se salvează
                              octetul care conține semnul primu-
                              lui operand ;
          MØV  A, L        ;
          SUB  E           ;  Z (=X−Y) → [H, L] ;
          MØV  L, A        ;
          MØV  A, H        ;
          SBB  D           ;
          MØV  H, A        ;
          XRA  B           ;
          RLC              ;  ZS⊕XS→CY ;
          JC   PØSDCR      ;  Dacă ZS⊕XS=1 este posibil DCR ;
CONTINUA: DCX  SP          ;  (SP0)+2→SP ;
          DCX  SP          ;
          PUSH H ;         ;  [ZS, Z1 ... Z7]→M<(SP0)+1>
                              și [Z8 ... Z15]→M<(SP0)> ;
                              (SP0)→SP ;
          DCX  SP          ;  (SP0)−2→SP ;
          DCX  SP          ;
          RET              ;  Revenirea din subprogram ;
```

```
PØSDCR:   MØV A, D      ;|
          XRA B         ;} YS⊕XS→CY;
          RLC           ;|
          JNC CØNTINUA  ;  Dacă YS⊕XS=0 nu este DCR cu
                           toate că ZS⊕YS=1;
          XTHL          ;
          LXI H, ADRDCR ;| FFH→M<ADRDCR>; se mar-
                         } chează
          MVI M, FFH    ;| apariţia DCR;
          XTHL          ;
          JMP CØNTINUA.
```

6.2.3. SUBPROGRAM DE ÎNMULŢIRE A DOI OPERANZI CU SEMN, DE CÎTE 16 BIŢI, REPREZENTAŢI ÎN COD COMPLEMENTAR FAŢĂ DE 2

Fie X şi Y cei doi operanzi iar $Z=X*Y$ rezultatul înmulţirii.

Se presupune că în momentul apelării subprogramului (printr-o instrucţie CALL) cei doi operanzi trebuie să se afle deja introduşi în stivă astfel :

	Adresa de memorie	Conţinutul memoriei	
Vîrful stivei	(SP_0)	Y8	Y15 Înmulţitorul
	$(SP_0)+1$	YS Y1z.....	Y7
	$(SP_0)+2$	X8	X15 Deînmul-
	$(SP_0)+3$	XS X1	X7 ţitul

Înmulţirea $Z=X*Y$ se va efectua conform metodei „Booth-Booth" [3] algoritmul urmat constînd din următoarele etape :

a) se citesc din stivă cei doi operanzi şi se introduc în registrele microprocesorului, deînmulţitul se introduce în [*B*, *C*] iar înmulţitorul în [*D*, *E*];

b) se anulează registrele [*H*, *L*] în care se constituie produsul şi bitul CY care prelungeşte înmulţitorul spre dreapta ;

c) se detectează şi se semnalizează apariţia *DCR* (acesta poate apare doar cînd $X=8\,000\ H$ şi $Y=8\,000\ H$) ;

d) se inspectează perechea formată din bitul cel mai puţin semnificativ al registrului înmulţitorului, E_0, şi bitul suplimentar CY care prelungeşte spre dreapta acest registru ;

— dacă cei doi biţi sînt identici se face deplasarea aritmetică cu o poziţie spre dreapta a registrelor pentru produs şi înmulţitor considerate în prelungire (deplasarea aritmetică este deplasarea în care se păstrează bitul de semn, în cazul considerat primul bit din stînga al produsului) ;

— dacă cei doi biți considerați [E_0, CY] au valoarea 01 se adună deînmulțitul cu produsul parțial, rezultatul înscriindu-se în registrele [H, L] pentru produs, iar apoi se face deplasarea aritmetică spre dreapta a registrelor pentru produs și înmulțitor considerate în prelungire ;

— dacă perechea celor doi biți considerați are valoarea 10 se scade deînmulțitul din produsul parțial, iar apoi se face deplasarea aritmetică spre dreapta a registrelor produsului și înmulțitorului considerate în prelungire.

Etapa *d)* se repetă de 16 ori, ultima dată nemaifăcîndu-se însă deplasarea aritmetică spre dreapta.

e) se înscrie în stivă rezultatul înmulțirii, rezultat în registrele [H. L, D, E], sub forma :

	Adresa de memorie	Conținutul memoriei	
Vîrful stivei →	(SP_0)	$Z24$ ..... $Z30$ * *	Octetul cel mai puțin semnificativ al produsului Z.
	(SP_0)+1	$Z16$ ..... $Z23$	
	(SP_0)+2	$Z8$ ..... $Z15$	
	(SP_0)+3	ZS $Z1$ ..... $Z7$	Octetul cel mai semnificativ al produsului Z.

Apariția sau neapariția *DCR* este indicată de subprogram prin înscrierea valorii *FFH* sau respectiv *00H* la adresa *ADRDCR* rezervată în memorie pentru semnalizarea *DCR*.

În continuare se prezintă subprogramul *DMULC2* (pentru efectuarea înmulțirii) scris în limbaj de asamblare 8080 :

```
DMULC2 : INX  SP          ;  (SP0)→SP
         INX  SP          ;
         LXI  H, ADRDCR   ;  Se inițializează locația de memorie des-
         MVI  M, 00H      ;  tinată semnalizării DCR; 00H→M◁
                          ;  <ADRDCR>;
         PØP  D           ;  Înmulțitorul Y→[D, E] : [YS, Y1 ...
                             ...Y7]→D, [Y8... Y15]→E; (SP0)+
                             +2→SP;
         PØP  B           ;  Deînmulțitul X→[B, C] : [XS, X1 ...
                             ... X7]→B, [X8 ... X15]→C; (SP0)+
                             +4→SP;
         MØV  A, B        ;  Dacă [XS, X1 ... X7]≠1,0 ... 0 nu
         CPI  80H         ;  poate apare DCR și se face salt la eti-
         JNZ  NØDCR       ;  cheta NØDCR;
         MØV  A, C        ;  Dacă [X8 ... X15]≠0 ... 0 nu poate
         CPI  00H         ;  apare DCR și se face salt la NØDCR;
         JNZ  NØDCR       ;
```

```
        MØV A, D        ;|  Dacă [YS, Y1 ... Y7]≠1, 0 ... 0 nu
        CPI 80H         ;>  poate apare DCR şi se face salt la
        JNZ NØDCR       ;|  NØDCR;
        MØV A, E        ;|  Dacă [Y8 ... Y15]≠0 ... 0 nu poate
        CPI 00H         ;>  apare DCR şi se face salt la NØDCR;
        JNZ NØDCR       ;|
        LXI H, ADRDCR   ;|  Deoarece în acest punct X=1,0 ..., 0
        MVI M, FFH      ;|  şi Y=1,0 ... 0, apare DCR
        PUSH B          ;|  (SP0)→SP;
        PUSH D          ;|
        JMP FINAL       ;
NØDCR:  MVI H, 10H      ;|  16 D→CØNTOR; se iniţializează con-
        PUSH H          ;>  torul de cicluri şi se introduce în stivă
                        ;|  care are acum vîrful (SP)=(SP0)+2;
        LXI H, 0000H    ;   Se anulează registrele [H, L] pentru
                            produs
        XRA A           ;   Se anulează CY care se va considera ca
                            bit de prelungire spre dreapta a înmul-
                            ţitorului
CICLU:  IC ADR1         ;   Dacă (CY)=1 se sare la eticheta ADR1
        MØV A, E        ;|  Dacă (E0)=0 se sare la eticheta ADR2
        RAR             ;>  deoarece (CY) este deja=0;
        JNC ADR2        ;|
        MØV A, L        ;|
        SUB C           ;|  Deoarece cei doi biţi E0 şi CY au valoa-
        MØV L, A        ;>  rea 10, deînmulţitul se scade din pro-
        MØV A, H        ;|  dusul parţial;
        SBB B           ;|
        MØV H, A        ;|
        JMP ADR2        ;
ADRL:   MØV A, E        ;|  Dacă şi E0 are valoarea 1 se sare la
        RAR             ;>  ADR 2;
        JC ADR 2        ;|
        DAD B           ;   Deoarece în acest punct cei doi biţi
                            inspectaţi E0 şi CY au valoarea 01
                            deînmulţitul se adună la produsul par-
                            ţial;
ADR2:   XTHL            ;|
        MØV A, H        ;|  Dacă s-a ajuns la ultimul ciclu se sare
        CPI 01H         ;>  peste deplasarea aritmetică la dreapta;
        XTHL            ;|
        JZ ADR 3        ;|
```

```
        MØV A, H        ;\
        RLC             ; |  Se face deplasarea aritmetică spre
        RRC             ; |  dreapta a registrelor [H, L, D, E] şi
        RAR             ; |  a bitului CY considerate toate în
        MØV H, A        ; |  prelungire (conţinînd produsul parţial
        MØV A, L        ; |  şi biţii neinspectaţi încă ai înmulţito-
        RAR             ; |  rului);
        MØV L, A        ; |
        MØV A, D        ; |
        RAR             ; |
        MØV D,A         ; |
        MØV A,E         ; |
        RAR             ; |
        MØV E,A         ;/
ADR3:   XTHL            ;\
        DCR H           ; |  CØNTØR=CØNTØR-1;
        XTHL            ;/
        JNZ CICLU
        XTHL  \
              |  [ZS, Z1 ... Z7]→M<(SP0)+3>;  [Z8 ... Z15]
              |  →M<(SP0)+2>;
        PUSH  |  D; [Z16 ... Z23]→ M<(SP0)+1>;  [Z24 ... Z28]
              |  →M<(SP0)>; (SP0); (SP0)→SP
FINAL:  DCX SP          ;\  (SP0)-2→SP;
        DCX SP          ;/
        RET             ;   Revenirea din subprogram.
```

6.2.4. SUBPROGRAM DE ÎMPĂRŢIRE A DOI OPERANZI CU SEMN, DE CÎTE 16 BIŢI, REPREZENTAŢI ÎN COD COMPLEMENTAR FAŢĂ DE 2

În cele ce urmează se va nota cu X deîmpărţitul, cu Y împărţitorul, cu Z cîtul iar cu R restul acestei împărţiri ($X=Z*Y+R$).

Se presupune că în momentul apelării subprogramului de împărţire (printr-o instrucţie *CALL*) deîmpărţitul şi împărţitorul se află deja introduşi în stivă, conform următoarei convenţii :

	Adresa de memorie	Conţinutul memoriei	
Vîrful stivei →	(SP_0)	Y8 ..... Y15	Împărţitorul
	$(SP_0)+1$	YS Y1 ..... Y7	
	$(SP_0)+2$	X8 ..... Y15	Deîmpărţitul
	$(SP_0)+3$	XS X1 ..... X7	

Pentru realizarea împărțirii se folosește metoda „fără refacerea restului" [3]. Algoritmul care se implementează cuprinde următoarele etape :

a) cei doi operanzi se citesc din stivă și se introduc în registrele microprocesorului ; deîmpărțitul introducîndu-se în [*B*, *C*] iar împărțitorul în [*D*, *E*]. Se anulează registrele [*H*, *L*] în care se asamblează biții cîtului.

b) se stabilește bitul de semn al cîtului : acesta va fi egal cu XS dacă $YS=0$ sau va fi egal cu $\overline{XS}$ dacă $YS=1$. Bitul de semn se introduce în L_0 (rangul cel mai puțin semnificativ al perechii de registre [*H*, *L*] în care se constituie cîtul).

c) se verifică posibilitatea de efectuare a împărțirii ; pentru aceasta ;

— se adună împărțitorul la deîmpărțit dacă $XS \neq YS$ sau se scade împărțitorul din deîmpărțit dacă $XS=YS$;

— dacă $(D_7)=0$ și $(B_7)=(L_0)$ sau dacă $(D_7)=1$ și $(B_7)\neq(L_0)$ sau dacă $(D_7)=1$ și $(B_6 \ldots B_0, C_7 \ldots C_0)\neq 0$ nu se poate efectua împărțirea și se semnalizează *DCR* ;

d) se calculează cifrele curente ale cîtului parcurgîndu-se următoarele etape :

— se deplasează cu o poziție spre stînga registrele [*B*, *C*, *H*, *L*] considerate în prelungire ;

— dacă $(L_1)=0$ se adună împărțitorul (aflat în [*D*, *E*]) la conținutul registrelor [*B*, *C*] iar dacă $(L_1)=1$ se scade împărțitorul din conținutul acestor registre ;

— dacă $(D_7)=0$ se înscrie $(\overline{B_7})$ în L_0 iar dacă $(D_7)=1$ se înscrie (B_7) în L_0 ;

Se repetă etapa *d)* de 15 ori

e) se corectează cîtul conținut în registrele [*H*, *L*] prin adunarea la conținutul acestor registre a unei unități de ordinul cel mai puțin semnificativ ;

f) se înscrie în stivă rezultatul împărțirii (cîtul Z din registrele [*H*, *L*] și restul R din registrele [*B*, *C*]) :

	Adresa de memorie	Conținutul memoriei	
Vîrful stivei →	(SP_0)	$R8$ ..... $R15$	Restul : $R*2^{15}$
	$(SP_0)+1$	RS $R1$ ..... $R7$	
	$(SP_0)+2$	$Z8$ ..... $Z15$	Cîtul : Z
	$(SP_0)+3$	ZS $Z1$ ..... $Z7$	

Apariția sau neapariția *DCR* este indicată de subprogram prin înscrierea valorii *FFH* sau respectiv *00H* la adresa *ADRDCR* rezervată în memorie pentru semnalizarea *DCR*.

În continuare se prezintă subprogramul *DDIVC2* (pentru efectuarea împărțirii) scris în limbaj de asamblare 8080 :

```
DDIVC2 : INX SP      ;\  (SP0)→SP ;
         INXSP       ;/
```

```
        LXI H, ADRDCR   ;\  0→M<ADRDCR>;
        MVIM, 00H       ;/
        PØP D           ;   Împărțitorul Y→[D, E]; [YS,
                            YL ... Y7]→D; [Y8 ... Y15]→E;
                            (SP0)+2→SP;
        PØP B           ;   Deîmpărțitul X →[B, C]; [XS,
                            X1 ... X7]→B; [X8 ... X15]→C;
                            (SP0)+4→SP;
        LXI H, 0000H    ;\  0→[H, L]; (CØNTØR=0)→Stivă;
        PUSH H          ;/  (SP0)+2→SP;
        MØV A,D         ;\  Dacă YS=1 se sare la eticheta ADR 10
        RLC             ;   avînd totodată [XS, X1 ... X7] in-
        MØV A, B        ;   trodus în A;
        JC ADR 10       ;/
        RAL             ;\  Deoarece YS=0 se introduce XS în L0;
ADR11:  MØV A, L        ;
        RAL             ;
        MØV L, A        ;/
        MØV A, B        ;\  Dacă XS⊕YS=0 se sare la ADR 20;
        XRA D           ;
        JP ADR20        ;/
        MØV A,C         ;\
        ADD E           ;   Deoarece XS⊕YS=1, adică XS≠
        MØV C,A         ;   ≠YS, se adună împărțitorul la de-
        MØV A, B        ;   împărțit
        ADC D           ;
ADR21:  MØV B, A        ;/
        MØV A, D        ;\  Dacă (D7)=0 se sare la ADR30;
        RLC             ;
        JNC ADR30       ;/
        MØV A, B        ;\  Dacă (B7)≠(L0) în condițiile în care
        RLC             ;   (D7)=1 apare DCR și se sare la
        XRA L           ;   ADR40, unde se tratează acest caz;
        RAR             ;
        JC ADR40        ;/
        MØV A, B        ;\  Dacă [B6 ... B0] 0≠chiar condițiile
        ANI 7FH         ;   în care (D7)=0, nu poate apare DCR;
        JNZ CICLU       ;/
        MØV A, C        ;\  Dacă [C7 ... C0]=0 în condițiile în
        JZ ADR40        ;   care (D7)=0 și [B6 ... B0]=0 apare
                            DCR și se sare la eticheta ADR40;
```

```
CICLU    DAD H          ;\
         MØV A, C       ;  Deplasarea spre stînga a registrelor
         RAL            ;  [B, C, H, L] considerate în prelungire;
         MØV C, A       ;/
         MØV A, B       ;\
         RAL            ;
         MØV B, A       ;/
         MØV A, L       ;\  Dacă (L1)=1 se sare la eticheta
         ANI 02H        ;/  ADR50;
         JNZ ADR50      ;
         MØV A, C       ;\  Deoarece (L1)=0 împărțitorul aflat
         ADD E          ;  în [D, E] se adună la conținutul
         MØV C, A       ;/ registrelor [B, C];
         MØV A, B       ;\
         ADC D          ;
ADR51 :  MØV B, A       ;/
         MØV A, D       ;\  Dacă (D7)=0 se sare la ADR60;
         RLC            ;/
         MØV A, B       ;\
         JNC ADR60      ;/
ADR61 :  ELC            ;\
         ANI 01H        ;  Dacă (D7)=1 se înscrie (B7) în L0;
         ØRA L          ;/
         MØV L, A       ;
         XTHL           ;\  CØNTØR=CØNTØR+1;
         INR H          ;  Dacă CØNTØR<15 se reia ciclul de
         MØV A, H       ;/
         CPI 0F H       ;  determinare a biților cîtului;
         XTHL           ;
         JNZ CICLU      ;
         INX H          ;  Se corectează cîtul Z conținut în [H, L]
         XTHL           ;  cîtul Z→[M<(SP0)+3>, M<(SP0)
                           +2>]
         PUSH B         ;  Restul R→[M<(SP0)+1>, M<
                           <(SP0)>]; (SP0)→SP
FINAL :  DCX SP         ;\  (SP0)−2→SP;
         DCX SP         ;/
         RET            ;  Revenirea din subprogram;
ADR10 :  RAL            ;\  Deoarece YS=1 se pregătește XS
         CMC            ;  pentru a fi introdus în L0;
         JMP ADR11      ;/
```

```
ADR20 :  MØV A, C      ;
         SUB E         ;
         MØV C, A      ;
         MØV A, B      ;
         SBB D         ;
         JMP ADR21     ;
```

Deoarece $XS \oplus YS = 0$ adică $XS = YS$, se scade împărțitorul din deîmpărțit;

```
ADR30 :  MØV A, B      ;
         RLC           ;
         XRA L         ;
         RAR           ;
         JNC ADR 40 ;  ;
         JMP CICLU     ;
```

Deoarece $(D_7)=0$ se testează dacă și $(B_7)=(L_0)$ caz în care apare *DCR* și se sare la eticheta *ADR*40;

Dacă nu apare *DCR* se trece la *CICLU*;

```
ADR50 :  MØV A, C      ;
         SUB E         ;
         MØV C, A      ;
         MØV A, B      ;
         SBB D         ;
         JMP ADR51     ;
```

Deoarece $(L_1)=1$ împărțitorul aflat în $[D, E]$ se scade din conținutul registrelor $[B, C]$;

```
ADR60 :  RLC           ;
         CMA           ;
         JMP ADR61     ;
```

Deoarece $(D_7)=0$ se pregătește $\overline{(B_7)}$ pentru a fi înscris în L_0:

```
ADR40 :  LXI H, ADRDCR ;
         MVI M, FFH    ;
```

Deoarece a fost detectat *DCR* se face marcarea adresei de memorie destinată acestui scop : $FFH \rightarrow M<ADRDCR>$

```
         DCX SP        ;
         DCX SP        ;
         JMP FINAL     ;
```

$(SP_0) \rightarrow SP$;

BIBLIOGRAFIE

1. Rony P. R.; Larsen D. G., Titus J. A.: *The 8080 Bugbook. Microcomputer interfacing and programming*, Howard W. Sons § Co. Inc.-1977.
2. I. Dancea: *Microprocesoare. Arhitectură internă, programare, aplicații*, Ed. Dacia, Cluj-Napoca, 1979.
3. V. Pop: *Bazele logice ale calculatoarelor* — curs, partea I-a, Institutul Politehnic „Traian Vuia" Timișoara, 1972.

CAPITOLUL 7

APLICAȚII ÎN ROBOTICĂ

7.1. INTRODUCERE

Robotica reprezintă o categorie de aplicații ale tehnicii de calcul avînd un înalt grad de complexitate. Multă vreme construcția roboților a fost limitată de viteză și capacitatea de memorare a unei singure unități de calcul fie aceasta minicalculator sau un sistem de calcul, fiind în plus și foarte costisitoare.

Cel dintîi avantaj pentru robotică oferit de microprocesoare este că, odată cu reducerea volumului util ocupat și prețul scăzut, a devenit economică utilizarea unui număr mare de calculatoare în cadrul unui singur robot. Un număr de microprocesoare mici dar puternice pot fi folosite pentru a crea dispozitive senzoriale și de comandă inteligente alcătuind împreună subsisteme cu o coordonare unică la nivel de sistem [1].

Abordarea aplicațiilor industriale ale microprocesoarelor pentru construirea roboților, așa cum se întîmplă și în alte domenii, nu se poate face decît prin studierea și cercetarea principiilor roboticii adică cu roboți în laborator [2]. Uneori aceste sisteme de robotică ajung la o complexitate ridicată cuprinzînd în afara părților mecanice, senzori ultrasonici, tactili, laser, camere de televiziune și unități de calcul ierarhizate incluzînd microprocesoare, minicalculatoare și un sistem de calcul de mare capacitate (roboți integrali). [3].

În acest capitol se prezintă o aplicație a microprocesorului 8080 la realizarea unității de comandă pentru un robot mobil numit P.E.T.R.I.C.A. (de la PROIECTUL EXPERIMENTAL TIMIȘOREAN DE ROBOT INTELIGENT PENTRU CERCETARE AVANSATĂ).

7.2. ROBOTUL MOBIL P.E.T.R.I.C.A.

7.2.1. CONCEPȚIA DE ANSAMBLU

Obiectivul principal urmărit a fost construirea unui robot comandat de microprocesor care să servească în mod eficient unor lucrări de cercetare robotică. Alegerea soluției de robot mobil s-a făcut din două motive : 1) capacitatea de a furniza o varietate de probleme într-un cîmp larg cuprinzînd percepția, învățarea, luarea deciziilor etc. 2) posibilitatea de a realiza un sistem modular care se poate dezvolta treptat oferind aspecte multiple de cercetare la fiecare pas. Termenul robot mobil înseamnă că are atît mecanisme manipulatoare cît și mecanisme locomotorii acestea din urmă conferind calități de vehicul.

Fi. 7.1. Vedere generală a robotului mobil PETRICA (foto)

Robotul PETRICA arătat în figura 7.1 cuprinde pe partea mecanică un cărucior și un braț mecanic, a căror conducere se realizează de către o unitate de comandă cu microprocesor *INTAL* 8080 A și periferice proprii. Partea mecanică are ca model de referință robotul de laborator descris în [4].

S-a prevăzut posibilitatea conectării microprocesorului la un minicalculator pentru realizarea în viitor a unui sistem de robotică dotat cu senzori.

7.2.2. CĂRUCIORUL

Partea cu funcție de vehicul a robotului PETRICA se înfățișează ca un cărucior pe patru roți. Roțile din față sînt antrenate independent de două motoare pas cu pas (pas unghiular 15^0, putere 6,4 W) în timp ce roțile din spate de tip castor, se orientează liber.

Căruciorul are trei nivele constructive (vezi figura 7.2. a). Nivelul cel mai de jos, bază, adăpostește dispozitivele de locomoție printre care motoare pas cu pas, reductoare, surse de putere. Nivelul intermediar torsul, cuprinde toate sistemele electronice de comandă și sursele de alimentare ale acestora. Deasupra

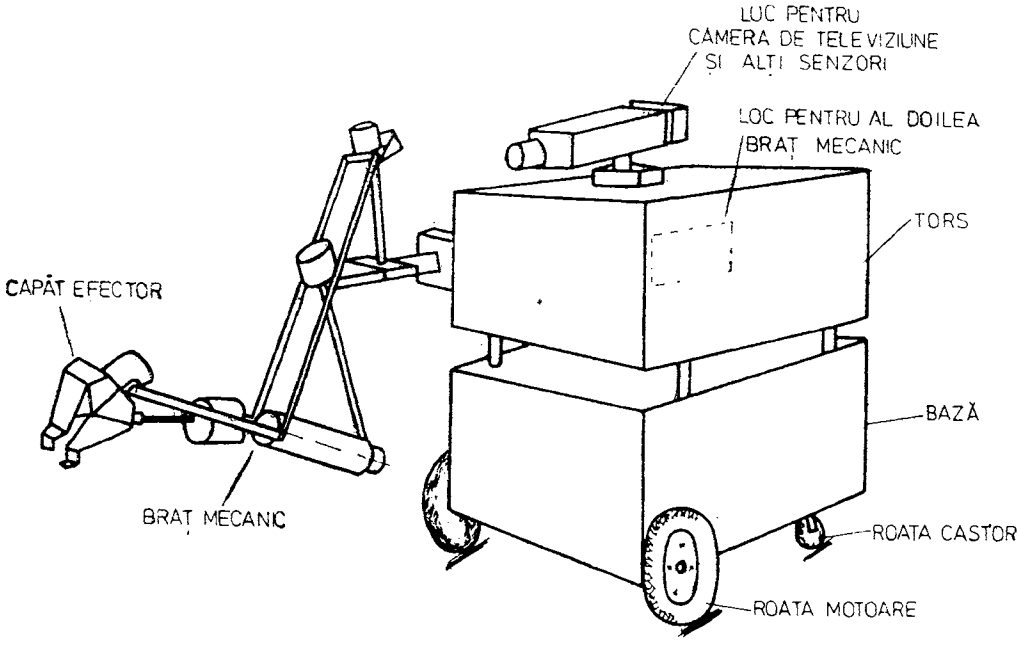


Fig. 7.2. Principalele elemente constructive

acestuia pe nivelul superior, se poate monta o cameră de televiziune și este prevăzut suficient spațiu pentru alte categorii de senzori ca, detectoare de proximitate ca ultrasunete, sisteme de triangulație în infraroșu și de radar, telemetru cu laser, ș. a.

Pe partea laterală dreaptă a torsului se află fixat un braț mecanic. Un al doilea braț mecanic identic cu primul poate fi montat pe partea stîngă dacă se dorește efectuarea unor experimente de „colaborare" între două brațe mecanice.

Prin combinații între sensurile celor două motoare de antrenare a roților sau prin acționarea unei roți și menținerea celeilalte frînate se obțin opt mișcări diferite (tab. 7.1). Astfel prin acționarea simultană în același sens a ambelor

Tabelul 7.1

Mișcările căruciorului obținute prin combinații între sensuri de mișcare și motoare

MIȘCAREA	MOTOR 7	MOTOR 8	SENS
ROTIRE STÎNGA	DA	NU	
ROTIRE DREAPTA	NU	DA	
ROTIRE PIRUETA	DA	DA	DIFERIT
DEPLASARE LINIARĂ	DA	DA	ACELAȘI

motoare se realizează mișcarea de DEPLASARE liniară ÎNAINTE sau ÎNAPOI indicată prin (A_{16}) în figura 7.3. În timpul mișcării acesteia căruciorul se deplasează pe o traiectorie liniară.

Mișcarea (9) numită ROTIRE PIRUETĂ, se deosebește de cea dinainte prin sensul diferit în care sînt acționate roțile. Această mișcare face ca robotul să se rotească în jurul unei axe virtuale care trece prin mijlocul distanței între roțile din față, punct notat cu *R*.

În fine prin acționarea separată a uneia din roți și menținerea celeilalte în stare frînată se execută mișcările notate (7) și (8) denumite ROTIRE STÎNGA și ROTIRE DREAPTĂ.

Pentru activitatea robotului ce implică mișcări pe sol ROTIRILE DR/ST sînt avantajoase pentru mișcarea de apropiere treptată a brațului de obiectul care urmează să fie manipulat. Mișcarea de tip ROTIRE PIRUETĂ este deosebit de utilă pentru deplasarea rapidă a obiectelor apucate pe distanțe mijlocii.

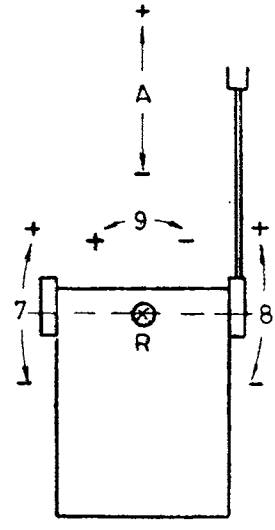


Fig. 7.3. Mișcările posibile ale căruciorului

7.2.3. MANIPULATORUL

Construcția manipulatorului este arătată în figura 7.4. Motoarele pentru acționare de tip pas cu pas, sînt plasate în articulații. Cele cinci grade de libertate (*GDL*) sînt : rotirea și ridicarea brațului (R_1 și P_1), ridicarea și rotirea antebrațului (P_2 și R_2), înclinarea mîinii (P_3). Al șaselea motor asigură închiderea și deschiderea mîinii adică mișcarea degetelor D_1 și D_2 dar acestea nu se consideră grad de libertate (Fig. 7.5).

Mișcarea cea mai largă se obține din umăr care permite rotirea brațului cu 90° la dreapta și la stînga adică 180° în total. Asemănător poate fi rotit și antebrațul. Această mișcare este potrivită pentru turnarea lichidelor, în timp ce rotația din umăr este utilă în cazul colaborării cu un alt braț.

Combinații între cele trei pivotări P_1, P_2 și P_3 permit ridicarea sau coborîrea mîinii și orientarea acesteia față de sol. Fiecare din cele trei pivotări realizează 45°.

Cele șase motoare electrice pas cu pas sînt identice cu cele folosite pentru locomoție. Utilizarea motoarelor de acest tip se știe, permite funcționarea în buclă deschisă și cunoașterea exactă a fiecărui unghi de articulație.

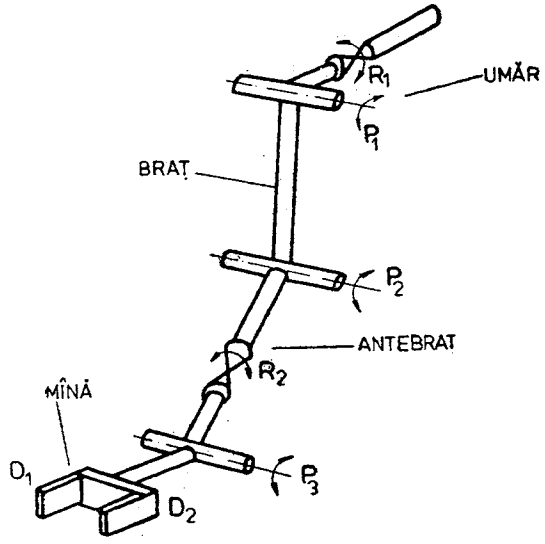


Fig. 7.4. Structura cinematică a manipulatorului

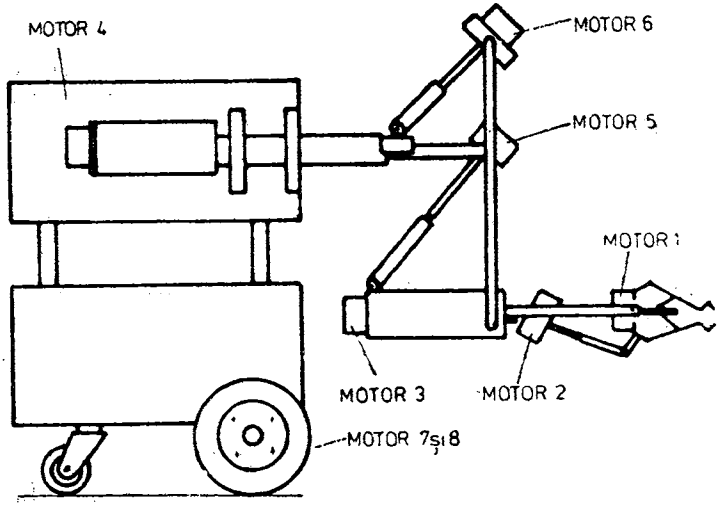


Fig. 7.5. Amplasamentul și numerotarea motoarelor

7.2.4. CONFIGURAȚIA SISTEMULUI DE COMANDĂ

Robotul mobil PETRICA posedă un sistem de comandă realizat în jurul microsistemului de dezvoltare care a constituit obiectul cap. 3. Se reamintește că acesta cuprinde microcalculatorul de dezvoltare și toate perifericele împreună cu interfețele respective adică ; tastatura *TO—H*, *DAF*1001 și casetofonul (conectate prin circuitul 8251), lectorul și perforatorul de bandă (Fig. 7.6).

Cele 8 motoare pas cu pas ale robotului sînt acționate de cîte un bloc electronic de comandă. Fiecare bloc de comandă a *MPP* are o schemă identică, bazată pe distribuitorul de impulsuri (fig. 4.27) și cheile de curent (fig. 4.32) descrise în capitolu 4.

Pentru ca un motor să execute un pas, la intrarea SENS trebuie stabilit nivelul logic corespunzător sensului de mișcare dorit (1 logic pentru sens direct și 0 logic pentru sens invers). Momentul declanșării pasului se face pe tranziția $1 \rightarrow 0$ a impulsului de tact adus la intrarea *ICDA*.

Generarea semnalelor $SENS_{1-8}$ se obține prin încărcarea registrului de ieșire de 8 bit SENS, cu vectorul binar pregătit în acumulator.

În privința impulsurilor $ICDA_{1-8}$ lucrurile sînt ceva mai complicate. Ele sînt furnizate la ieșirile registrului de 8 bit TACT, prin extragerea alternativă a doi vectori binari, unul avînd toți biții zero, celălalt avînd 1 logic doar în rangurile corespunzînd motoarelor ce trebuiesc acționate. Se presupune de exemplu că motoarele 1, 3, 7 și 8 vor executa un pas, primele două în sens direct iar ultimele două în sens invers. Rezultă că vectorul binar *TACT* va avea valoarea $C5_{16}$ iar vectorul SENS va lua valoarea 03_{16}. În acest caz secvența de program pentru generarea semnalelor *ICDA* va fi :

```
; SE PRESUPUNE CA REGISTRUL TACT A FOST INIȚIALIZAT
  PE ZERO
PASM:  MVI A, 03H   ;  SE PREGĂTEȘTE VECTOR SENS
       STA SENS        SE EXTRAGE VECTOR SENS
       MVI A, 0C5H  ;  SE PREGĂTEȘTE VECTORUL TACT
       STA TACT     ;  SE EXTRAGE VECTORUL TACT
       XRA A        ;  SE ANULEAZĂ ACUMULATORUL
       STA TACT     ;  SE ANULEAZĂ REGISTRUL TACT
            .
            .          ÎN ACEST MOMENT SE EXECUTĂ
            .       ;  PASUL MPP
```

Pentru aducerea brațului în poziția inițială sistemul de comandă dispune de un port de intrare prin care poate testa starea microcomutatoarelor de inițializare. Motorul articulației care nu se află în poziția de referință este acționat pînă în momentul în care se închide contactul microcomutatorului aflat pe articulația respectivă (0 logic).

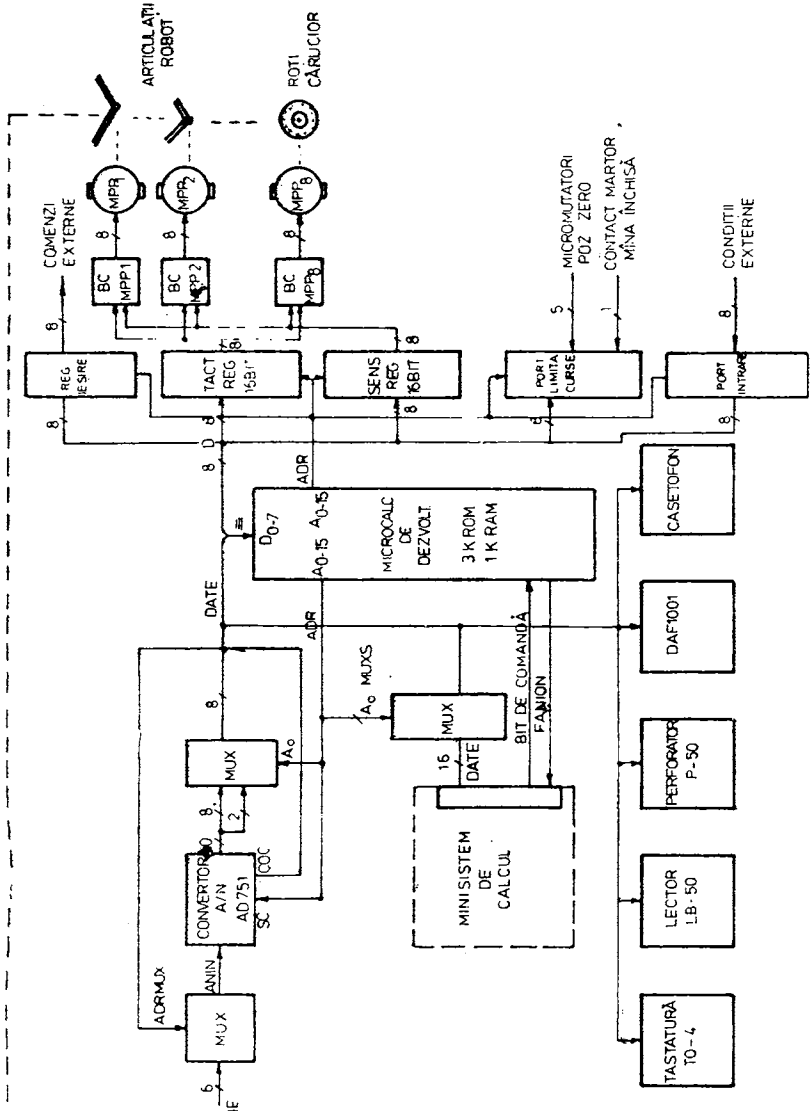


Fig. 7.6. Configurația sistemului de comandă cu microprocesor

Prin același port de intrare se poate citi starea logică a microcomutatorului care sesizează momentul închiderii mîinii. Se subliniază că acest microcomutator poate servi și pentru sesizarea prezenței unui obiect în mînă. Această din urmă posibilitate se explică prin faptul că degetele mîinii sînt antrenate printr-un servomecanism cu *MPP* care permite controlul deplin asupra poziției degetelor.

Dacă microcomutatorul este apăsat și degetele se află într-o poziție intermediară înseamnă că un obiect este prezent între degete.

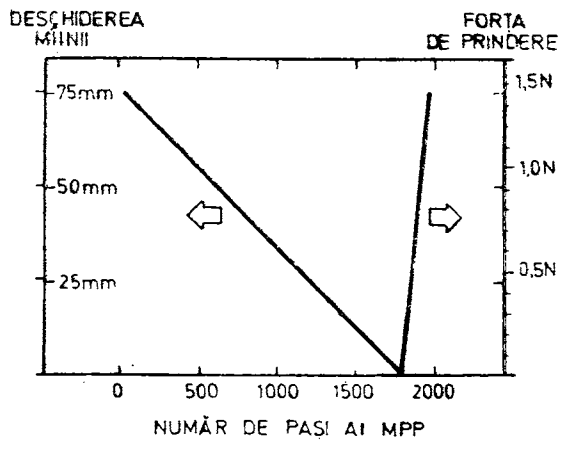


Fig. 7.7. Diagrama forței de prehensiune și deschiderii mîinii

Servomecanismul mîinii și motorul pas cu pas permit reglarea forței care se aplică obiectului manipulat conform cu diagrama din fig. 7.7. în care sînt reprezentate forța de prehensiune F și distanța între degete D_1D_2 în funcție de numărul de pași efectuat de motor.

În vederea sincronizării activității robotului cu alte echipamente cu care acesta conlucrează, sînt prevăzute două porturi, unul de ieșire și altul de intrare pentru generarea unor comenzi în exterior sau sesizarea unor condiții exterioare de tipul „tot sau nimic".

Pentru experimente de robotică în care anumite tipuri de senzori furnizează mărimi analogice, sistemul de comandă al robotului dispune de un modul de achiziții de date compus dintr-un multiplexor static pentru 16 canale și un convertor A/N pe 10 bit cu viteza de conversie de 25 microsecunde. Blocul de achiziții de date este obiectul unei descrieri separate.

Tot în cazul unor experimente mai complexe sistemul de comandă al robotului este prevăzut cu un port de intrare bidirecțional pentru 16 bit și cu semnale de comandă corespunzătoare pentru a fi conectat la un minicalculator [5].

Întreg sistemul de comandă este implementat pe patru plăci mari aflate într-un sertar îmbarcat pe robot. Interconexiunile cu perifericele sînt realizate fizic așa cum se observă în fig. 7.8. Lungimea cablurilor spre robot permite robotului să se deplaseze față de punctul fix de conexiune cu cîțiva metri.

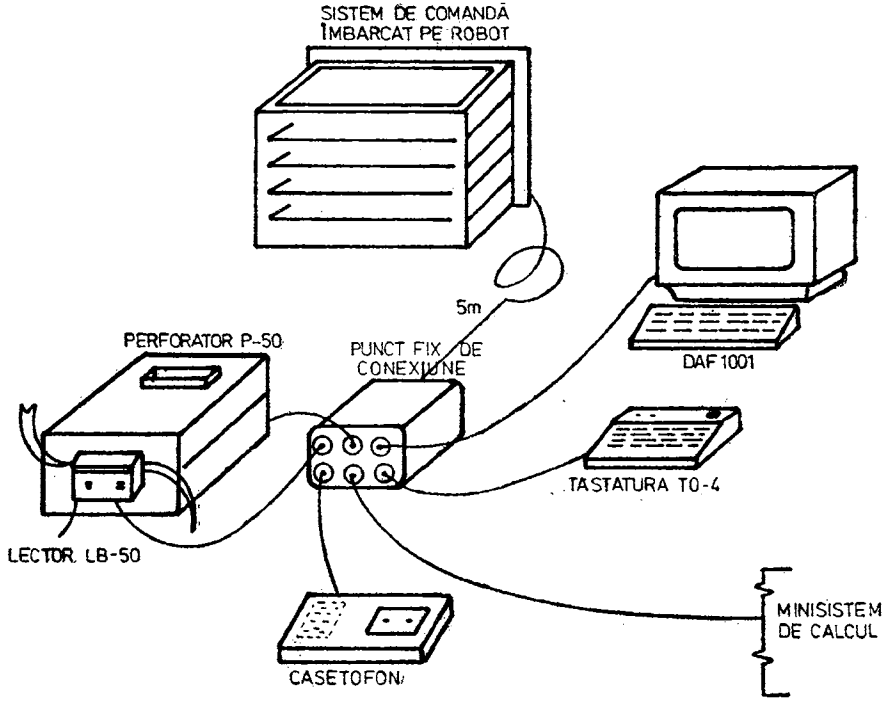


Fig. 7.8. Interconexiunile fizice ale robotului cu perifericele

Un aspect esențial în funcționarea robotului îl constituie programele cu care este echipat. În general sistemul de comandă al oricărui robot îndeplinește un număr de activități (task-uri) variate printre care :

— supervizarea sistemului ;
— transferul de date cu perifericele (consolă, discuri ...) ;
— comanda mașinilor și uneltelor;
— comanda tuturor articulațiilor ;
— dialogul cu operatorul ;
— etc.

Cîteva din programele care realizează aceste funcțiuni și care intră în dotarea robotului PETRICA au fost deja descrise în cap. 3. Ele sînt subprogramele de perforare, de citire a benzilor, de decodificare a comenzilor date de la tastatură TO-4.

Alte programe printre care cel de implementare a algoritmului de comandă a MPP prin accelerare și frînare și cel de învățare vor fi prezentate în continuare în paragrafele 7.3 și 7.4.

7.2.5. MODULUL DE ACHIZIȚII DE DATE

Schema convertorului și multiplexorului este arătată în figura 7.9. Se folosește un circuit integrat *AD* 571 care conține în aceeași capsulă circuitul pentru tensiunea de referință, generator de tact, comparator, registru pentru aproximarea succesivă (*SAR*) și registru tampon. Performanțele de bază sînt 10 bit și 25 de microsecunde timpul pentru o conversie. Pentru realizarea conversiei și funcției *SAR*, circuitul *AS* 571 întrebuințează cea mai avansată tehnologie de fabricație a circuitelor, I^2L. Rezistențe sînt realizate pe film subțire de SiCr de înaltă finețe. Compensarea cu temperatura se face cu o diodă Zener integrată. [6].

Funcționarea lui *AD* 571 în regim de conversie comandată prin impuls are loc astfel : cînd B § $\bar{C}$ devine „0" logic, ieșirile trei stări ale datelor sînt flotante și începe conversia. Terminarea ei este semnalată de $\bar{D}\bar{R}$ care ajunge la „0" și tot acum sînt validate ieșirile de date.

Semnalul pentru intrarea B § $\bar{C}$ este format de bistabilul *RS* implementat cu două porți 400. În urma executării unei instrucții de ieșire se generează semnalul *I /OW* și ADRCAN semnalul de selecție a convertorului. Aceste două semnale provoacă aducerea la intrarea B § $\bar{C}$ a lui „1" logic. După 1,5 μs semnalul $\bar{D}\bar{R}$ devenit „1" logic (răspunsul convertorului la cererea de conversie) aduce bistabilul la zero. Acum de abia începe conversia care durează aproximativ 25 μs. Revenirea la „0" a lui $\bar{D}\bar{R}$ marchează terminarea conversiei și poate fi sesizată prin program. Rezultatul conversiei se obține pe 10 bit care pot fi citiți printr-o singură instrucție *LHLD*.

Multiplexarea primilor 8 bit mai puțin semnificativi și a celor doi rămași pe magistrala de date a microprocesorului este asigurată de circuitele CDB 403 cu colector în gol validate de semnalele $CITLO = I/OR.\ ADRCAN.\bar{A}$ (0) și $CITHI = I/OR.ADRCAN.A$ (0). Tensiunea necunoscută este aplicată la intrarea *ANIN* (*ANALOGIN*).

Amplificatorul operațional μA 715 funcționează ca repetor și realizează adaptarea de impedanță între intrarea convertorului *ANIN* și ieșirea multiplexorului analogic. Rezistența semireglabilă de 50 ohmi protejează circuitul *AD* 571 la supracurenți. Pe de altă parte curentul absorbit de amplificator de la ieșirea multiplexorului este foarte mic datorită impedanței de intrare ridicate.

Așa cum se vede din figură schema mai cuprinde multiplexorul analogic, circuitele de translație *TTL-MOS* și logica de selecție a canalului. Multiplexorul analogic servește la selectarea tensiunilor din punctele de măsură și aducerea la intrarea *ANIN* a convertorului. Pentru selecția celor 16 canale se folosesc trei capsule integrate *MN* 2009*A*. Selecția unui canal se face prin aplicarea unei tensiuni de −15 V la una din intrările de selecție ale circuitului *MN* 2009 și +15 V la toate celelalte intrări. Valoarea tensiunii selectate va apare la ieșirea multiplexorului, căderea de tensiune pe tranzistorul MOS în regim de conducție fiind neglijabilă.

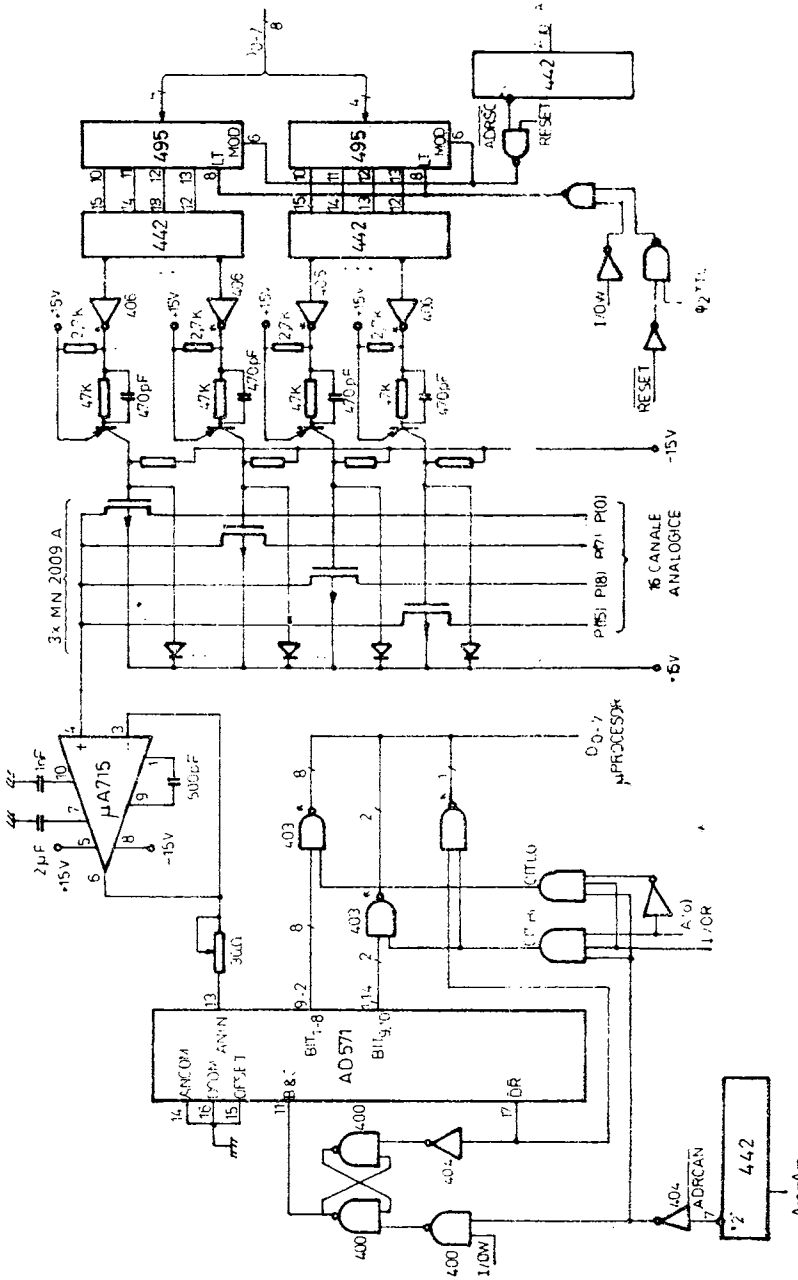


Fig. 7.9. Schema modulului de achiziții de date.

Tensiunile +15 V și −15 V sînt obținute prin translația nivelelor *TTL* în nivele *MOS*. Circuitul translator se compune dintr-un inversor cu colectorul în gol (*CDB* 406), un tranzistor de tip *pnp*, *BC* 251 și rezistențe de polarizare. Cînd la intrarea inversorului se aplică „0" logic tranzistorul este blocat și căderea de tensiune pe rezistența legată la +15 V este foarte mică ($Ico.R_1$), insuficientă pentru a deschide tranzistorul. Tranzistorul fiind blocat la ieșire în colectorul tranzistorului se obține −15 V. Rezultă că selectarea unui canal se face prin „0" logicl Tranzistorul se saturează cînd la intrarea inversorului se aplică „1" logic.

Selecția unui canal din cele 16 posibile se face înscriind un octet în registrul de selecție (*SC*) implementat cu două capsule *CDB* 495. Fiecare decodificator CDB 442 realizează selecția pentru 8 canale analogice. Codul de selecție este încărcat cu octetul transmis pe magistrala de date prin comenzile $LT=RESET.\ \emptyset_{2TTL}+I/OW$ și $MOD=RESET+ADRSC$.

În operația de selectare a canalului octetul transmis în registrul *SC* trebuie să fie de forma XF_{16} sau FX_{16}, unde X este o cifră hexazecimală de la 0 la 7. Tabelul 7.2 arată corespondența fiecărui cod de selecție cu canalul selectat. La inițializare (*RESET*) în registrul *SC* se încarcă *FF* deci nici un canal nu va fi selectat.

Din punctul de vedere al programării operațiile de selecție a canalului dorit și declanșare a conversiei se desfășoară ca în secvența din tabelul 7.3. La capătul acestei secvențe se află instrucția *LHLD ADRCAN* care are meritul că permite citirea a doi octeți (unul incomplet) de la adresele *ADRCAN*

Tabelul 7.2

Coduri de selecție a decanalelor analogice

CANAL	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
1	0	0	0	0	1	1	1	1
2	0	0	0	1	1	1	1	1
3	0	0	1	0	1	1	1	1
4	0	0	1	1	1	1	1	1
5	0	1	0	0	1	1	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	0	1	1	1	1
8	0	1	1	1	1	1	1	1
9	1	1	1	1	0	0	0	0
10	1	1	1	1	0	0	0	1
11	1	1	1	1	0	0	1	0
12	1	1	1	1	0	0	1	1
13	1	1	1	1	0	1	0	0
14	1	1	1	1	0	1	0	1
15	1	1	1	1	0	1	1	0
16	1	1	1	1	0	1	1	1

Tabelul 7.3.

Subprogram de citirea poziției unei articulații

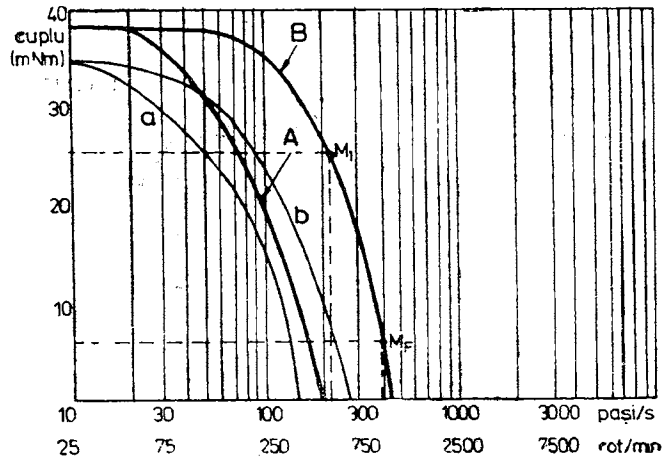
```
;  SUBPROGRAM CARE ASIGURĂ SELECTAREA POTENȚIOMETRULUI ȘI CANA-
;  LULUI DORIT APOI DECLANȘEAZĂ CONVERSIA ANALOG-NUMERICĂ ȘI INTRO-
;  DUCE CEI 10 BIT REPREZENTÎND POZIȚIA ARTICULAȚIEI
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   ADRSC  EQU        8400H     ;  ADRESA SELECTOR CANAL
   ADCAN  EQU        8800H     ;  ADRESA CONVERTOR A/N
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   START: MVI A,     CANAL     :  SE ÎNCARCĂ COD CANAL
          STA        ADRSC     :  START CONVERSIE ANALOG
                               ;  NUMERICĂ
   TEST:  LDA        ADCAN        ÎNCARCĂ BITUL SF. CONVERSIE
          RRC                  ;  SE TESTEAZĂ SFÎRȘITUL
          JC         TEST      ;  CONVERSIEI
          LHLD                 ;  SE ÎNCARCĂ CEI 10 BIT ÎN H ȘI L
          SHLD       POZIT     ;  SE DEPUN ÎN RAM LA ADRESELE
                               ;  POZIT ȘI POZIT +1
```

(8800_{16}) și *ADRCAN*+1 (8801) pe care îi depune în registrele *L* și *H*. În felul acesta se asigură transferul cel mai rapid cu putință între convertor și microprocesor.

7.3. ALGORITMUL DE COMANDĂ AL MOTOARELOR PAS CU PAS

Pentru mișcările robotului PETRICA se folosesc în total opt motoare pas cu pas repartizate astfel : două pentru cărucior, cinci pentru brațul mecanic și unul pentru mînă.

În caracteristicile motorului pas cu pas (fig. 7.10) se remarcă existența curbei START-STOP care indică pentru un anumit cuplu motor frecvența de comandă la care rotorul rămîne în sincronism la pornirea din repaus sau la oprire (nu apare pierdere de pași sau depășire). Curba a doua de FUNCȚIONARE, indică frecvențele maxime cu care poate fi comandat motorul fără pierdere de pași. Cele două curbe și zonele delimitate sînt suportul alegerii punctului sau punctelor succesive de funcționare ale *MPP*.



- linia subțire fără rețea RC
- linia groasă cu rețea RC
- A, a caracteristica START-STOP
- B, b caracteristica de FUNCTIONARE

Fig. 7.10. Caracteristicile motorului pas cu pas cu 4 înfăşurări (seria ID 08 -Phillips)

Pentru obţinerea unor mişcări line ale articulaţiilor braţului, la pornire şi oprire este nevoie de realizarea unei accelerări treptate, respectiv decelerări înainte de oprire [7]. Din punctul de vedere al motorului aceasta înseamnă o lege de variaţie pentru frecvenţa de comandă a MPP astfel încît să se permită deplasarea punctului de funcţionare din zona START-STOP în cea de FUNCŢIONARE şi înapoi în mod treptat. Forma de variaţie a frecvenţei de comandă în funcţie de timp şi numărul de paşi care să permită accelerarea, funcţionarea la viteză constantă şi frînare, este arătată în fig. 7.11 [8].

Exemplu : la una din articulaţiile robotului, motorul pas cu pas antrenează un şurub avînd pasul 1 mm astfel încît pentru 100 de paşi piuliţa se va deplasa cu circa 4,16 mm. Servomecanismul prezintă un moment de frecare $M_f=6$ mNm şi un moment inerţial total (rotor+sarcină+şurub), $M_{JT}=125\times10^{-4}$ gm^2.

Dacă viteza impusă este de 100 paşi în 0,5 s rezultă o frecvenţă de comandă de v=200 p/s.

Motorul care se alege trebuie să aibă un cuplu mai mare de 6 mNm la o frecvenţă de pas mai mare decît 200 p/s.

Pe caracteristica de FUNCŢIONARE a motorului pas cu pas cu patru înfăşurări s-a determinat o viteză maximă considerînd numai cuplul de frecare.

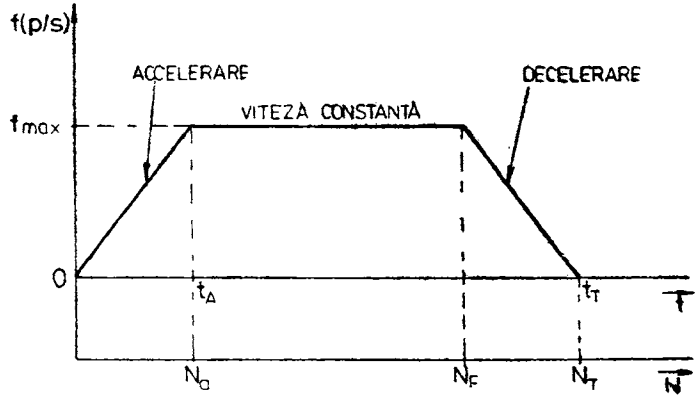


Fig. 7.11. Diagramă de mișcare a MPP cu controlul accelerației

O a doua determinare mai exact estimare a frecvenței de lucru (mai mică decît cea maximă), 230 p/s conduce la diferența :

$$M_1 - M_F = 28 - 6 = 22 \text{ mNm.}$$

Cu un coeficient de siguranță de 50% se obține :

$$22 \text{ mNm} \times 0{,}5 = 11 \text{mNm.}$$

Folosind relația :

$$M = J_M \times \frac{V}{t} \boxtimes K, \text{ unde } K = \frac{2 \times \pi}{\text{nr. pași/rev.}} = \frac{2 \times \pi}{24} = 0{,}26,$$

se obține valoarea $t = t_A$, adică timpul de accelerare :

$$t_A = \frac{125 \times 10^{-4} \times 230 \times 0{,}26}{11} = 0{,}068.$$

Deoarece timpul de accelerare t_A este egal cu cel de decelerare t_D se poate determina numărul de pași pentru accelerare și decelerare :

$N_A + N_D = \frac{V}{2} \times t_A \times 2$ sau,

$N_A + N_D = v \times t_A = 230 \times 0{,}068 = 16$ pași.

Timpul de mișcare la frecvența maximă este în acest caz :

$$T_{FUNCT.} = \frac{N_T - (N_A + N_D)}{V} = \frac{100 - 16}{230} = \frac{84}{230} = 0{,}360 \text{ s,}$$

în care $N_T=100$, numărul total de paşi. Verificarea timpului total pentru mişcare se face însumînd :

$T_{FUNC}+t_A+t_D$, sau

$0{,}360+0{,}068+0{,}068=0{,}5$ s.

Exemplul de calcul de mai sus este o primă estimare. Se poate face mişcarea mai înceată dacă se doreşte o siguranţă mai mare sau mai rapidă dacă se urmăreşte optimizarea ei.

Algoritmul pentru comanda motorului pas cu pas pentru cazul discutat 8 paşi pentru accelerare, 8 pentru frînare este următorul :

. dacă $N_T>16$ se aplică comanda de accelerare (primii 8 paşi), viteza maximă şi apoi frînare (ultimii 8 paşi) ;

. dacă $N_T\leqslant 16$ se comandă motorul cu o viteză fixă la o frecvenţă de sincronism mult mai mică decît cea maximă.

Subprogramul care implementează acest algoritm este arătat în tab. 7.4. şi fig. 7.12. El compară mai întîi numărul total de paşi pe care trebuie să-l efectueze motorul, cu valoarea 16 şi decide dacă mişcarea se va face cu frec-

Tabelul 7.4.

Subprogram de comandă a motorului pas cu pas cu accelerare şi frînare

```
;] SUBPROGRAMUL PERMITE ACCELERAREA (PE PRIMII 8 PAŞI), VITEZA CON-
;  STANTĂ ŞI FRÎNAREA (PE ULTIMII 8 PAŞI) SE POT EXECUTA UN NUMĂR DE
;  65 536 PAŞI, LA VITEZA MAXIMĂ DE 200 p/s. NUMĂRUL DE PAŞI SE TRANS-
;  MITE PRIN REGISTRUL B ŞI C
;  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  ADTAB  EQU      2300H      ;  ADRESA BAZĂ A TABELULUI CU ÎNTÎR
                                ZIERI
   * * * * * * * * *** * * * * * * * * * * * * * * * * * * * * * * * * * * *
;  CDMPP : MOV      A, B       ;  SE TRANSFERĂ B ÎN ACUMULATOR
          CPI      00H        ;  SE COMPARĂ CU ZERO
          JNZ      ACCEL      ;  DACĂ NU E ZERO SALTĂ ACCELERARE
          MOV      A, C       ;  SE ADUCE C ÎN ACUMULATOR
          CPI      10H        ;  NR. PAŞI SE COMPARĂ CU 16
          JP       ACCEL      ;  DACĂ NR. PAŞI>16 SALT ACCELERARE
          RZ                  ;  DACĂ NPPAŞI=0 REVINE
;  SE COMANDĂ MPP LA VITEZA CONSTANTĂ DE 50 P/S
;  CONST : LXI      D, 0600H   ;  ÎNCARCĂ VALOAREA ÎNTÎRZIERII
          CALL     PAS        ;  DECLANŞEAZĂ UN PAS
          CALL     TIMP       ;  ÎNTÎRZIERE 20 MS
          DCX      B          ;  MINUS UNU ÎN B ŞI C
          MOV      A, B       ;  TRANSFER B ÎN A
          ORA      C          ;  SAU LOGIC CU C
          JNZ      CONST      \ ;  DACĂ REZULT. NU E ZERO
          RET                 / ;  CONTINUĂ. DACĂ NU REVINE
```

Tabelul 7.4. (continuare)

```
;
; SE ACCELEREAZĂ MOTORUL PAS CU PAS
ACCEL : LXI     H, ADTAB ; SE ÎNCARCĂ ADR. TABELULUI
        MOV     C, M     ; SE ADUNĂ UNU ÎN L
        INX     L        ; SE ADUCE OCTETUL L DIN TABEL
        MOV     D, M     ; SE ADUNĂ UNU ÎN L
        INX     L        ; SE ADUCE OCTETUL 2 DIN TABEL
        CALL    PAS      SE DECLANȘEAZĂ UN PAS
        CALL    TIMP     ; ÎNTÎRZIERE VARIABILĂ
        DCX     B        ; MINUS UNU ÎN B ȘI C
        MOV     A, L     ; SE TRANSFERĂ L ÎN ACUMULATOR
        CPI     OEH      ; SE COMPARĂ L CU 14
        JNZ     ACCEL    ; DACĂ L NU E 14 SE ACCELEREAZĂ ÎN
                         ; CONTINUARE
SE MENȚINE VITEZA MAXIMĂ DE 200 P/S
VMAX:   CALL    PAS      ; SE DECLANȘEAZĂ UN PAS
        CALL    TIMP     ; ÎNTÎRZIERE 20 MS
        DCX     B        ; MINUS UNU ÎN B SI C
        MOV     A, B     ; SE TRANSFERĂ B ÎN A
        CPI     00H      ; SE COMPARĂ CU ZERO
        JNZ     VMAX     ; DACĂ NU MENȚINE VITEZA
        MOV     A, C     ; SE TRANSFERĂ C ÎN A
        CPI     07 H     ; AU MAI RĂMAS 7 PAȘI ?
        JP      VMAX     ; DACĂ DA SE COMANDĂ FRÎNA.
; SE FRÎNEAZĂ MOTORUL PÎNĂ LA OPRIRE
FRÎNA : DCR     L        ; SE MICȘOREAZĂ L CU UNU
        MOV     D, M     ; SE ADUCE OCTETUL ?
        DCR     L        ; SE MICȘOREAZĂ L CU UNU
        MOV     C, M     ; SE ADUCE OCTETUL 1
        CALL    PAS      ; SE DECLANȘEAZĂ UN PAS
        CALL    TIMP     ; ÎNTÎRZIERE VARIABILĂ
        DCX     B        ; SE MICȘOREAZĂ CU UNU
        MOV     A, B     ; SE TRANSFERĂ B ÎN A
        ORA     C        ; SAU LOGIC CU C
        JNZ     FRÎNĂ    ; MAI SÎNT PAȘI ?
        RET              ; SE REVINE ÎN PRG. PRINCIPAL
```

venţa fixă de 50 p/s sau cu variaţia frecvenţei de comandă. Partea interesantă a subprogramului este modul în care se realizează creşterea şi descreşterea vitezei motorului, de la frecvenţa 0 Hz pînă la 200 Hz viteza maximă, în 8 paşi. Deoarece primul pas se dă cînd frecvenţa este teoretic 0 p/s, rămîn 7 paşi pentru a atinge viteza maximă. Acest lucru conduce la o pantă a acceleraţiei de f=200 $p/s/7p$=28,5 Hz şi la următoarele valori ale frecvenţei de comandă :

28,5 Hz ; 57 Hz, 85,5 Hz, 114 Hz, 142,5 Hz, 171 Hz şi 200 Hz

Este cunoscut deja din subprogramul din Tab. 4.2. destinat comenzii unui *MPP* că frecvenţa de comandă se stabileşte prin alegerea intervalului de timp între două impulsuri de pas. Acest timp (perioada) se realizează în cadrul unei bucle prin decrementarea într-un registru (sau pereche de registre). Valoarea iniţială care se găseşte în registru determină întîrzierea.

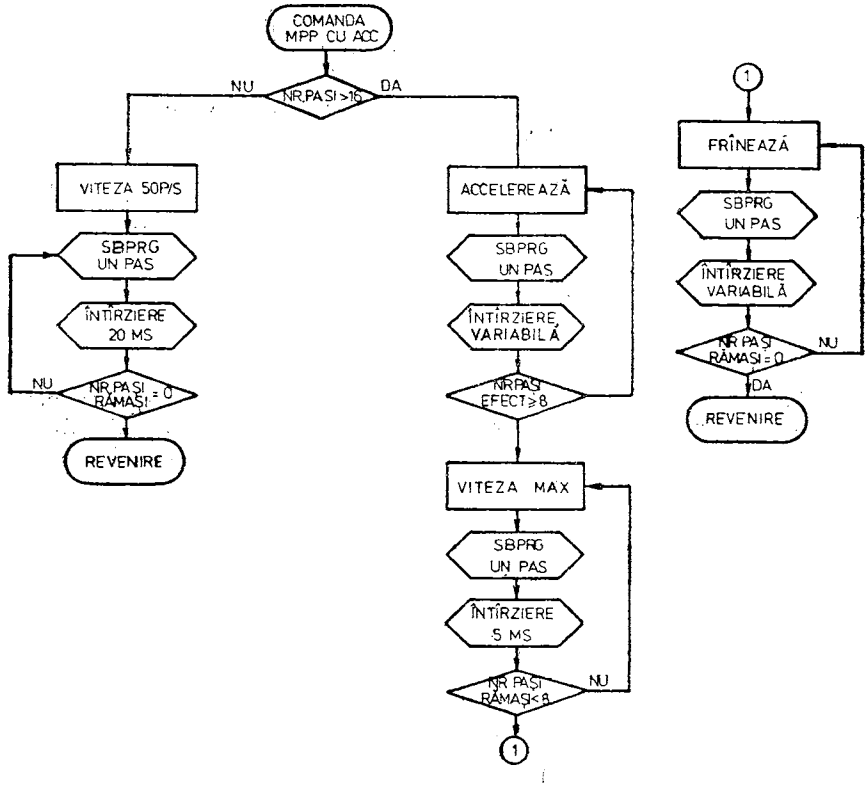


Fig. 7.12. Organigrama subprogramului de comandă cu accelerare a MPP.

În cazul de față treptelor de frecvență care apar în timpul accelerării le corespund următoarele întîrzieri : 35 ms, 17 ms, 11 ms, 8,7 ms, 7 ms, 5,8 ms și 5 ms. Din punct de vedere matematic aceste valori de timp se calculează cu formula $T = \frac{1}{n \times f}$, unde n este numărul de pași.

Introducerea acestei formule de calcul în programul care implementează algoritmul de comandă al *MPP* se justifică doar în situația în care se dorește modificarea parametrilor mișcării și se dipune de subprograme de calcul aritmetic eventual de un interpretator *BASIC* în varianta pentru microprocesoare (*MBASIC*) [9].

Tabelul 7.5.

Valorile întîrzierilor pentru accelerare şi decelerare

ÎNTÎRZIEREA	VALOAREA ZECIMALĂ		VALOAREA ÎN HEXA	
35	2 916	——→ACCELERARE	0B64	DECELERARE←——
17	1 416		0588	
11	916		0394	
8,7	725		02D5	
7	583		0248	
5,8	483		01E3	
5	416		01A0	

În absenţa acestor mijloace software există alternativa implementării funcţiei de accelerare, dînd programului de comandă valorile T sub formă tabelară (Tab. 7.5). Parcurgerea întîrzierilor de sus în jos sau de jos în sus permite accelerarea sau frînarea *MPP*.

7.4. EXPERIMENT DE IMPLEMENTARE A CONCEPTULUI DE PROGRAMARE PRIN INSTRUIRE

Avînd ca suport material robotul mobil PETRICA şi sistemul de comandă cu microprocesor descris, s-au putut iniţia experimente de robotică. Un astfel de experiment legat de programarea roboţilor este relatat în continuare.

Programarea roboţilor se poate face rigid (programe fixe), prin instruire de către un operator, sau în viitor prin instruire parţială restul fiind lăsat pe seama robotului care dispune de organe senzoriale şi programe de luare a deciziilor [10].

Conceptul de instruire este esenţial în robotică şi constituie deja o caracteristică majoră a roboţilor generaţiei a doua [11]. Se poate înţelege acest concept din figura 7.13 în care procesul de instruire şi redare sînt reprezentate schematic. Într-o primă etapă operatorul uman comandă mişcări ale robotului de la un pupitru. Dacă mişcările comandate corespund activităţii propuse, operatorul decide memorarea lor. Pas cu pas astfel de mişcări executate sub comanda şi observaţia directă a operatorului sînt incluse în viitorul program de lucru. Operatorul poate programa şi operaţii de testare a unor condiţii exterioare de tipul „tot sau nimic" pentru sincronizarea activităţii. În acelaşi scop

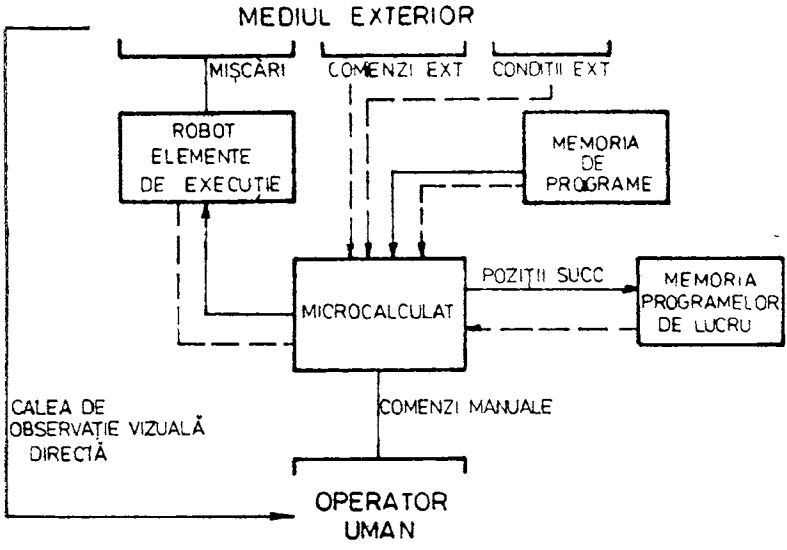


Fig. 7.13. Principiul de programare a roboţilor prin instruire

se pot genera comenzi pentru declanşarea unor operaţii în exterior de exemplu un pistol de sudură sau un pistol de vopsit. Durata unei astfel de comenzi poate fi programată tot de către operator prin aşa-numitele întîrzieri tehnologice [12].

Odată terminată instruirea se poate trece la verificarea şi apoi la reducerea automată a programului de lucru.

Dacă la cele două operaţii instruire şi redare adăugăm operaţia de iniţializare şi una de aşteptare a comenzilor operatorului rezultă patru stări diferite în total. De aceea activitatea robotului poate fi reprezentată sub formă de graf al stărilor unui automat finit (fig. 7.14).

Înainte de elaborarea programelor destinate unei funcţionări ca cea descrisă anterior trebuie parcursă etapa definirii structurii datelor.

În fiecare pas de lucru al robotului e nevoie de următoarele informaţii: poziţia nouă a articulaţiilor, sensul de mişcare, condiţiile externe, comenzile externe şi întîrzierile tehnologice. Toate aceste informaţii sînt reunite în cadrul unui BLOC de date cu lungimea fixă de 21 de octeţi (fig. 7.15). Bineînţeles modul în care s-a ajuns la reprezentarea fiecărei categorii de informaţii se cere explicat. Astfel, acţionarea fiecărui motor al robotului se face prin specificarea deplasărilor relative faţă de poziţia deţinută. Mai precis în cadrul celor doi octeţi atribuiţi fiecărui motor se dă în binar numărul de impulsuri de comandă cu care trebuie comandat motorul respectiv în cadrul pasului de program

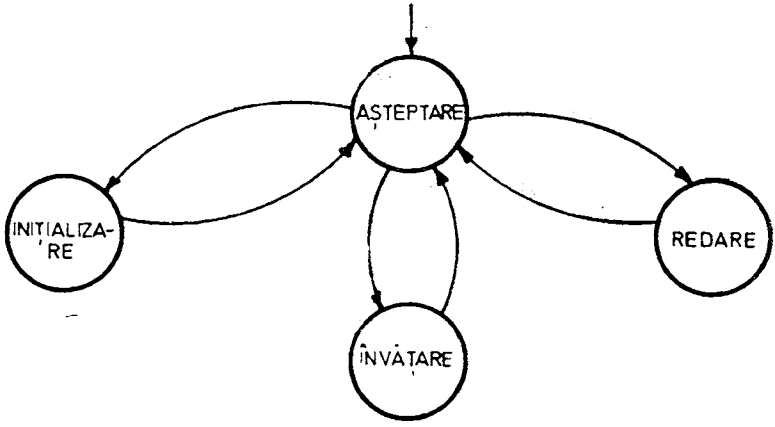


Fig. 7.14. Graful stărilor principale ale robotului

BLOC	BIT 7 ... BIT 0
0	DEPLASARE RELATIVĂ MOTOR 1
1	
2	DEPLASARE RELATIVĂ MOTOR 2
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	DEPLASARE RELATIVA MOTOR 8
15	
16	VECTOR TACT
17	VECTOR SENS
18	CONDITII EXTERNE
19	COMENZI EXTERNE
20	ÎNTÎRZIERI TEHNOLOGICE

Fig. 7.15. Blocul de date pentru un pas de program

(maximum 65 536) impulsuri într-un pas). Pentru cele opt motoare rezultă aşadar 16 octeţi. Întrucît generarea impulsurilor de comandă pentru *MPP* se face prin program înscriind „1" sau „0" în registrul *TACT*, în rangul motorului dorit, este avantajos ca vectorul de comandă care grupează tactele celor 8 motoare să fie deja pregătit încă din timpul instruirii. În acest fel se face economie de timp de procesor la execuţie. De asemeni pentru precizarea sensurilor de mişcare a fost prevăzut un octet (*SENS*). Convenţional s-a stabilit că + se reprezintă prin „1" logic şi semnul — prin „0" logic.

Din examinarea structurii blocului rezultă că în cadrul unui pas de program pot fi realizate orice fel de combinaţii de mişcări.

Testarea condiţiilor exterioare se execută dacă octetul prevăzut pentru aceasta este diferit de zero. Dacă de exemplu în rangul 5 se găseşte un „1" acest lucru va fi interpretat în sensul că nu se va trece la pasul următor pînă cînd semnalul exterior numărul 5 nu va fi „1" logic. Tot un octet este destinat generării de comenzi în exterior. Durata unei comenzi exterioare poate fi menţinută prin programarea unei întîrzieri tehnologice în cadrul octetului ÎNTÎRZIERI. Domeniul de prescriere este de la 0 la 256 s.

Tabelul 7.6.

Lista comenzilor de la tastatura T0-4 pentru operarea robotului

SIMBOL NOU AL TASTEI	SIMBOL VECHI AL TASTEI	FUNCŢIA TASTEI
+		PE TIMPUL APĂSĂRII SE EXECUTĂ MIŞCAREA ÎN SENSUL DIRECT
—	—	PE TIMPUL APĂSĂRII SE EXECUTĂ MIŞCAREA ÎN SENS INVERS
1	1	ACŢIONARE MOTOR 1
2	2	ACŢIONARE MOTOR 2
3	3	ACŢIONARE MOTOR 3
4	4	ACŢIONARE MOTOR 4
5	5	ACŢIONARE MOTOR 5
6	6	ACŢIONARE MOTOR 6
7	7	ACŢIONARE MOTOR 7
8	8	ACŢIONARE MOTOR 8
9	9	ACŢIONARE MOTOR 7 ŞI MOTOR 8 ÎN SENSURI DIFERITE
A	A	ACŢIONARE MOTOR 7 ŞI MOTOR 8 ÎN ACELAŞI SENS
C	C	ŞTERGEREA ZONEI DE DATE DESTINATĂ BLOCULUI DE DATE
P	TAB	PERFORAREA BLOCULUI DE DATE
I	.	INIŢIALIZAREA ROBOTULUI (BRAŢUL MECANIC ÎN POZ. DE REFERINŢĂ)
L	LIST	LECTURA BENZII PERFORATE CONŢINÎND PROGRAMUL DE LUCRU

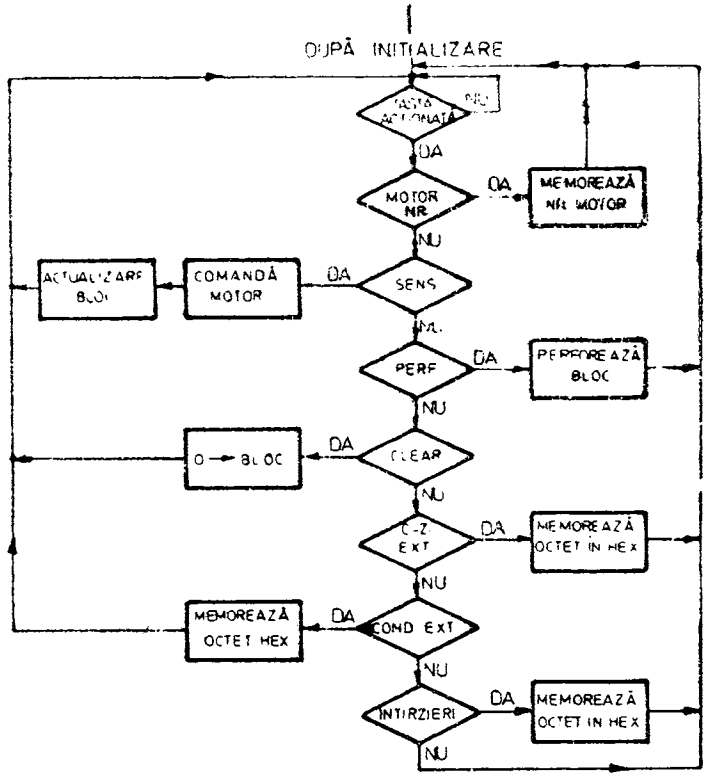


Fig. 7.16. Organigrama programului SCOLAR în timpul instruirii

Toate subprogramele care asigură funcționarea robotului în regim de redare sau instruire sînt grupate în cadrul programului ȘCOLAR care ocupă 630 de octeți în memoria *RAM* a microcalculatorului. Lansarea programului are loc de la adresa 2000_{16} adică prima locație din *RAM*. În prealabil programul este încărcat de pe bandă perforată prin citire la lector.

În linii mari funcționarea programului în cursul instruirii se petrece ca în figura 7.16. Orice acțiune este declanșată în urma detectării unei taste funcționale în cursul inspectării tastaturii. În tabelul 7.6 sînt prezentate tastele pe care programul le decodifică și execută. Se reamintește că aceeași tastatură TO-4 poate comanda funcții de bază ale microcalculatorului așa cum s-a văzut în subcapitolul 3.3. Acest lucru este valabil atîta timp cît se rulează programul MONITOR. În orice moment al funcționării robotului se poate trece la

programul MONITOR prin simpla apăsare a tastei *RESET*. Acesta reprezintă o ilustrare a procedeului de recalificare funcțională a unei tastaturi.

Comanda manuală pentru efectuarea unei mișcări se dă astfel : se acționează tasta cu numărul motorului 1 la 6 pentru braț și 7, 8, 9, A, pentru cărucior. Această comandă va fi memorată de ȘCOLAR. Declanșarea mișcării se petrece abia cînd se acționează tasta cu sensul dorit (+sau —). Mișcarea continuă atîta timp cît se menține apăsarea tastei. În orice moment se poate schimba sensul de mișcare dacă se dorește acest lucru de către operator.

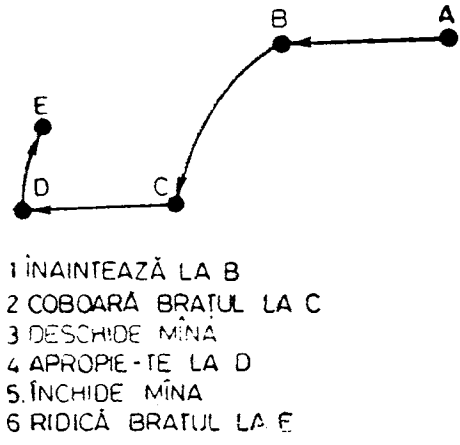


Fig. 7.17. Traiectoria mîinii robotului

Generarea unui pas de program se face printr-o ștergere prealabilă a zonei de memorie unde se depun datele din care se compune blocul pasului de program. Acest lucru are loc prin apăsarea tastei „C". Dacă s-a avut în grijă ca brațul să fie inițializat și căruciorul se află într-o poziție marcată pe sol, se poate trece la instruire.

O secvență scurtă de instruire a robotului pentru executarea unui grup de mișcări — înaintează, coboară brațul, deschide mîna, apropie-te, prinde cubul, ridică brațul, stop — este arătată în tabelul 7.7. și fig. 17.

Analizînd tastele acționate se observă desigur că în cadrul celui de-a doilea pas de program s-au prevăzut mișcări simultane de coborîre a brațului și antebrațului. După fiecare pas se comandă perforarea blocului de date generat pe bandă perforată. La capătul secvenței de instruire din tab. 7.7. se obține o bandă perforată avînd cinci blocuri. Formatul de perforare a blocului include un caracter marcă care precede și succede blocul. Mărcile sînt folosite la controlul citirii benzii la redare.

Redarea se desfășoară conform organigramei din figura 7.18. După introducerea benzii perforate pe care s-a înregistrat programul de lucru se poate declanșa executarea acestuia în mod automat prin apăsarea tastei „L". După citirea fiecărui bloc robotul trece la executarea operațiilor prevăzute prin instruire. Se începe bineînțeles cu interpretarea și executarea mișcărilor. Zonele de dublu octet care indică deplasările pentru fiecare motor sînt utilizate ca numărătoare cu decrementare. Oprirea unui motor are loc în momentul în care conținutul, celor doi octeți este zero și continuă în caz contrar pentru celelalte motoare.

Tabelul 7.7.

Exemplu de programare prin instruire funcția cerută descrierii efectului

C	ȘTERGEREA ZONEI DE DATE DESTINATE BLOCULUI	
A	DEPLASARE (MOTOR 7 ȘI 8)	ÎNAINTEAZĂ
+	CĂRUCIORUL SE DEPLASEAZĂ SPRE CUB	
P	PERFOREAZĂ BLOC	
C	ȘTERGEREA ZONEI DE DATE DESTINATE BLOCULUI	COBOARĂ BRAȚUL ȘI ANTEBRAȚUL
6	BRAȚ (MOTOR 6)	
—	COBORÎRE BRAȚ	
5	ANTEBRAȚ (MOTOR 5)	
—	COBORÎRE ANTEBRAȚ	
P	PERFORARE BLOC	
C	ȘTERGEREA ZONEI DE DATE DESTINATE BLOCULUI	
1	MÎNĂ (MOTOR 1)	DESCHIDE MÎNA
+	DESCHIDE MÎNA	
P	PERFORARE BLOC	
C	ȘTERGEREA ZONEI DE DATE DESTINATE BLOCULUI	
A	DEPLASARE (MOTOR 7 ȘI 8)	APROPIERE
+	DEPLASARE ÎNAINTE	
P	PERFORARE BLOC	
C	ȘTERGEREA ZONEI DE DATE DESTINATE BLOCULUI	
1	MÎNA (MOTOR 1)	PRINDE CUBUL
—	ÎNCHIDE MÎNA	
P	PERFORARE BLOC	
C	ȘTERGEREA ZONEI DE DATE DESTINATE BLOCULUI	
5	ANTEBRAȚ	RIDICĂ ANTEBRAȚUL
+	RIDICĂ ANTEBRAȚ	
P	PERFORARE BLOC	
SFÎRȘIT		

Dacă s-au epuizat toate mișcările programate se trece la testarea condițiilor externe și la emiterea comenzilor spre exterior în cazul cînd octeții prevăzuți sînt diferiți de zero.

În cele din urmă se verifică conținutul octetului pentru întîrzieri tehnologice și se apelează subprogramul pentru întîrzieri de o secundă de atîtea ori de cîte s-a specificat în pasul de program.

Prin aceasta execuția unui pas de program este terminată și poate continua cu pasul următor prin reluarea operațiilor arătate.

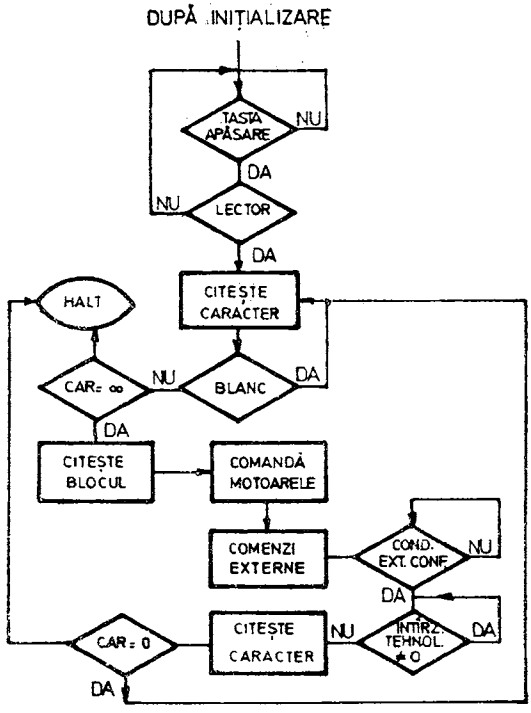


Fig. 7.18. Organigrama programului SCOLAR în timpul redării

Cu ajutorul programului ȘCOLAR s-au elaborat un număr de aplicații printre care construcții din cuburi aranjate într-o varietate de configurații, mișcarea unor piese pe tabla de șah, modelarea în laborator a programului unui robot industrial care deservește o linie de fabricație cu patru mașini-unelte.

BIBLIOGRAFIE

1. B. M. Dobrotin și D. A. Rennels: *An application of microprocessors to a Mars roving vehicle*, JACC, vol. 2, 1977.
2. R. Prajoux, *Robotics Research in France*, Robotics Age, Spring 1980, vol. 2, Nr. 1, pg. 16.

3. Ed. G. E. Pozdviak, *Integralnîe roboti*, Moskva, vol. 1 și 2 Mir 1971 și 1975.
4. * * * *Robotics*. Grivet Series 1979, GALLAHER RESEARCH, INC. 1979.
5. K. Goksel et al.: *An Intelligent Industrial Arm Using a Microprocessor*, IEEE Trans. Ind. Electron. and Contr. Instrum., vol. IECI — 22, pp. 309—314, Aug. 1975.
6. * * * *Data acquisition products catalog*, 1978, Analog Devices.
7. R. Paul, *The Mathematics of Computer Controlled Manipulators*, JACC, vol. 2, 1977, TA 11.
8. * * * *Stepper motor User's manual*, AIRPAX North American Philips Conteds Corp, 1980.
9. John W. Hill, *Mini Mover 5*, Robotics Age, Automn. vol. 2, No. 1.
10. M. Drăgănescu, *A doua revoluție industrială*, Editura Tehnică, București, 1980.
11. R. Stoinescu și V. Enătescu: *Principii psihocibernetice aplicate la roboții dotați cu inteligență senzoriomotoare*, Ed. Dacia, Cluj-Napoca, 1977.
12. * * * : *Manual for UNIMATE 4000*, Unimation Inc.

CUPRINS

Prefață 5

CAPITOLUL 1

Aspecte de interes general asupra operațiilor de intrare-ieșire

1.1. Transferul datelor prin program 8
1.2. Transferul datelor prin intermediul întreruperilor 9
1.3. Transferul datelor prin acces direct la memorie 10

CAPITOLUL 2

Realizarea unui sistem cu microprocesorul 8080. Varianta 1. 12

2.1. Calculul încărcării magistralelor de adresă și de date 13
2.2. Organizarea microcalculatorului 18
2.3. Selectarea blocurilor de memorie 18
2.4. Generarea semnalelor de comandă 21
2.5. Panoul de comandă 22
2.5.1. Încărcarea memoriei RAM de la panou 23
2.5.2. Regimuri de funcționare 25
2.5.3. Întreruperi generate de la panou 27
2.5.4. Tastatura hexazecimală 27
2.6. Modul de realizare a operațiilor de intrare-ieșire 30
2.7. Interfața cu miniimprimanta SEIKO 33
2.7.1. Schema interfeței cu miniimprimanta 34
2.7.2. Programele necesare tipăririi 37
2.8. Interfața cu dispozitivul de afișare DAF 1001 43
2.8.1. Programarea circuitului USART 8251 44
2.9. Interfața cu monitorul TV Tehnoton 48
2.9.1. Probleme generale privind modul de funcționare 48
2.9.2. Schema interfeței cu monitorul TV 51
2.10. Programator pentru memoria EPROM 2708 condus de microcalculator 57
2.10.1 Schema bloc a programatorului 58
2.10.2. Convertorul de nivele și baza de timp 59
2.10.3. Subprogram pentru programarea memoriei 2708 61
2.11. Programator autonom pentru memoria EPROM 2708 65

CAPITOLUL 3

Realizarea unui microcalculator : Varianta 2 72

3.1. Introducere 72
3.2. Microcalculatorul de dezvoltare 74
3.2.1. Concepția de bază 74
3.2.2. Configurația microcalculatorului 76
3.2.3. Circuitele de memorie 79
3.2.4. Magistralele de adresă și de date 80
3.2.5. Organizarea intrărilor și ieșirilor 81
3.2.6. Afișarea adresei, datelor și octetului de stare 83
3.2.7. Regimuri de funcționare 85
3.2.8. Conectarea tastaturii TO-4 88
3.2.9. Punerea la punct a microcalculatorului 90
3.2.10. Modul de operare 94
3.2.11. Programul MONITOR 98
3.3. Interfața cu lectorul de bandă LB-50 101
3.3.1. Prezentarea lectorului 101
3.3.2. Schema interfeței 102
3.3.3. Subprogramul de citire 105
3.4. Interfața cu perforatorul de bandă P-50 107
3.4.1. Prezentarea perforatorului 107
3.4.2. Schema interfeței 109
3.4.3. Subprogramul de perforare 112
3.5. Interfața serie asincronă realizată prin program 114
3.5.1. Introducere 114
3.5.2. Analiza funcțională 114
3.5.3. Circuitele interfeței 115
3.5.4. Subprogramul DAFIN 115
3.5.5. Subprogramul DAFEM 118
3.6. Înregistrarea datelor binare pe casetofon 119
3.6.1. Introducere 119
3.6.2. Descrierea interfeței 120
3.6.3. Înregistrarea 122
3.6.4. Redarea 123
3.7. Sistem pentru întreruperi multiple 124
3.7 1. Introducere 124
3.7.2. Schema circuitelor de întrerupere 127
3.7.3. Subprogramul de tratare a întreruperilor multiple 129

CAPITOLUL 4

Interfețe cu alte periferice numerice 132
4.1. Conectarea unei tastaturi fără electronică de codare 132
4.1.1. Probleme ale tastaturilor cu contacte mecanice 132

4.1.2. Interfațarea la microsistem a unei tastaturi fără electronică proprie ... 136
4.1.3. Program de monitorizare a tastaturii ... 138
4.2. Conectarea unui dispozitiv de afișare cu elemente în 7 segmente ... 144
4.2.1. Interfațarea microsistemului cu dispozitivul de afișare ... 146
4.2.2. Subprogramul de conversie a datelor din hexazecimal în codul 7 segmente ... 150
4.3. Conectarea unui calculator de birou cu imprimanta la un microsistem ... 153
4.3.1. Modul de utilizare a calculatorului de birou FELIX CE-129 T 153
4.3.2. Interfața dintre microprocesorul 8080 și calculatorul de birou cu imprimantă FELIX CE-129 T ... 156
4.4. Comanda unui motor pas cu pas ... 162
4.4.1. Introducere ... 162
4.4.2. Principiul de funcționare ... 163
4.4.3. Schema de comandă a MPP ... 168
4.4.4. Amplificatoarele de putere ... 171
4.4.5. Realizarea comenzii MPP prin proiectare logică programată 173

CAPITOLUL 5

Conectarea unor periferice analogice ... 177

5.1. Probleme generale ale achiziției de date ... 177
5.2. Metode de conectare a convertoarelor analog-numerice la microprocesor ... 183
5.2.1. Conectarea convertoarelor analog-numerice rapide ... 183
5.2.2. Conectarea convertoarelor analog numerice lente ... 184
5.2.3. Conectarea unui convertor analog-numeric de 10 biți ... 188
5.3. Probleme generale ale distribuției analogice a datelor numerice 191
5.4. Conectarea unui înregistrator în coordonate folosind două convertoare numeric-analogice ... 194
5.4.1. Conectarea convertoarelor numeric-analogice la microprocesor ... 194
5.4.2. Subprogramul pentru trasarea de grafice cu ajutorul înregistratorului în coordonate ... 198
5.4.3. Scrierea caracterelor alfanumerice cu ajutorul înregistratorului în coordonate ... 200
5.4.4. Digitalizarea curbelor desenate folosindu-se înregistratorul în coordonatele X-Y conectat la microsistem ... 211

CAPITOLUL 6

Implementarea unor funcţiuni aritmetice de bază 216

6.1. Reprezentarea datelor numerice 217
6.1.1. Formate de reprezentare 217
6.1.2. Coduri binare pentru reprezentarea datelor 220
6.2. Implementarea operaţiilor aritmetice elementare între operanzi reprezentanţi în cod complementar faţă de 2 220
6.2.1. Subprogram de adunare a doi operanzi cu semn de cîte 16 biţi reprezentaţi în cod complementar faţă de 2 220
6.2.2. Subprogram de scădere a doi operanzi cu semn de cîte 16 biţi, reprezentaţi în cod complementar faţă de 2 222
6.2.3. Subprogram de înmulţire a doi operanzi cu semn, de cîte 16 biţi, reprezentaţi în cod complementar faţă de 2 224
6.2.3. Subprogram de împărţire a doi operanzi cu semn, de cîte 16 biţi, reprezentaţi în cod complementar faţă de 2 227

CAPITOLUL 7

Aplicaţii în robotică 232

7.1. Introducere 232
7.2. Robotul mobil PETRICA 233
7.2.1. Concepţia de ansamblu 233
7.2.2. Căruciorul 233
7.2.3. Manipulatorul 235
7.2.4. Configuraţia sistemului de comandă 237
7.2.5. Modulul de achiziţii de date 241
7.3. Algoritmul de comandă al motoarelor pas cu pas 244
7.4. Experiment de implementare a conceptului de programare prin instruire 250

Redactor : GHEORGHE MANDICI
Tehnoredactor : IOAN I. IANCU

Bun de tipar : 26.10.1981.
Apărut 1981.
Coli tipar : 16,5.
şi una planşă.

Întreprinderea poligrafică „Banat"
Timişoara, Calea Aradului 1.
Comanda nr. 130.

ÎN EDITURA FACLA AU MAI APĂRUT

Dan Teodorescu :
MAŞINI ELECTRICE

Mircea Reghiş :
ELEMENTE DE TEORIA MULŢIMILOR
ŞI DE LOGICĂ MATEMATICĂ

Adrian Corduneanu :
ECUAŢII DIFERENŢIALE CU APLICAŢII
ÎN ELECTROTEHNICĂ

V. Obădeanu :
ELEMENTE DE ALGEBRĂ LINIARĂ
ŞI GEOMETRIE ANALITICĂ

Aurel Nanu, Dan Nanu :
PRELUCRAREA DIMENSIONALĂ PRIN EROZIUNE
ELECTRICĂ ÎN CÎMP MAGNETIC

P. Stanciu, D. Criveanu, Gh. David, W. Fuchs :
MATEMATICI APLICATE ÎN ECONOMIE

Eugen Pop, Vasile Stoica, Sever Crişan :
MĂSURĂRI ÎN ENERGETICĂ

Colectiv :
PROGRAMAREA CALCULATOARELOR

L. Vuc., C. Milici, I. Dimoiu, N. Rachin :
MODELE MATEMATICE
ÎN PROIECTAREA CONSTRUCŢIILOR

Al. Vasilievici, L. Moldovan :
ELEMENTE DE TEHNOLOGIE A APARATELOR
DE JOASĂ TENSIUNE

Gh. Savii :
LASERI

Lei 18,50